

On Solving Structured SAT on Ising Machines

A Semiprime Factorization Study

Ahmet Efe, Hüsrev Cilasun, Abhimanyu Kumar, Nafisa Prova, Ziqing Zeng, Tahmida Islam, Ruihong Yin, Chaohui Li, Peter Kreye, Chris Kim, Sachin S. Sapatnekar, Ulya R. Karpuzcu

University of Minnesota, Twin Cities

{efe00002, cilas001, kumar663, prova026, zeng0083, islam112, yin00473, li003135, krey0031, chriskim, sachin, ukarpuzc}@umn.edu

Abstract—Ising machines are emerging as a new technology for solving various classes of computationally hard problems of practical importance, yet their limits on structured SAT workloads – representative of numerous real-world applications – remain unexplored. We present the first systematic study of such problems, using semiprime factorization as a representative case. Our results show that highly restrictive, *tight* constraints, when mapped into optimization form, fundamentally distort Ising dynamics, and that these distortions are amplified when problems are decomposed to fit within limited hardware. We propose a hybrid approach that offloads constraint-heavy components to classical preprocessing while reserving the computationally challenging part for the Ising machine. Structured SAT represents a crucial step toward real-world applications, which remain out of reach today due to Ising machine limitations. Our findings reveal that constraint handling is a central obstacle and highlight hybrid hardware–software approaches as the path forward to unlocking the long-term potential of Ising machines. We conduct our evaluation on the manufactured Ising chips and demonstrate that our flow more than doubles the solvable problem size on a 45-spin all-to-all Ising chip, from 8-bit (94 variables) to 11-bit (190 variables), without hardware changes.

I. INTRODUCTION

Ising machines harness physical processes to solve computationally hard real-life problems of practical importance. By encoding logical or combinatorial tasks into energy minimization models, they provide a hardware-native pathway to optimization. As alternatives to traditional solvers, Ising machines offer speed and energy efficiency through direct exploitation of physics.

Recent work [1] has demonstrated that Ising machines can successfully solve *random* Boolean satisfiability (SAT) problems. SAT is a central problem with wide-ranging applications in verification, planning, and cryptography. Although SAT is computationally hard, random SAT problems, artificially crafted to benchmark solvers, often fail to capture the structure of real-world problems. In contrast, the ability of Ising machines to address structured SAT problems – derived from real-world applications – remains unexplored.

Structured SAT instances arise in domains such as arithmetic, hardware verification, and cryptography. Their defining feature is the presence of intricate variable dependencies and tight constraints, which often translate into highly correlated optimization rewards/penalties when mapped to an Ising machine. A particularly important example is semiprime

factorization, the foundation of RSA cryptography. In general, factorization refers to decomposing an integer into its prime factors. Semiprime factorization is the special and hardest case, where the integer is the product of exactly two primes, making it computationally challenging. We can obtain a highly structured SAT instance by generating a multiplication circuit to express the product of the primes, and by translating each of its gates into SAT form, using the standard Conjunctive Normal Form (CNF). Such SAT instances inherit the inherent difficulty of factoring through the highly inter-dependent constraints produced when arithmetic is expressed in SAT form. When the multiplier circuit is encoded as a SAT instance and translated into an Ising optimization task, the Ising machine searches for the lowest-energy state that corresponds to the integer factors of the semiprime. Note that the circuit output (the integer to be factored) is known in this case.

The security of modern cryptosystems such as RSA depends on the hardness of factorization, which may be compromised or reinforced by emerging technologies including Ising machines. This work investigates the ability of Ising machines to solve structured SAT problems, using semiprime factorization as a representative case study. We identify the core challenges introduced by tight arithmetic constraints, propose preprocessing techniques tailored to Ising machines, and evaluate problem decomposition strategies for scaling to larger factorization instances. *Our results provide the first systematic analysis of how Ising machines interact with structured SAT, highlighting both their current limitations and the methods that can extend their reach.*

II. BACKGROUND

Ising machines are physics-inspired hardware solvers for numerous computationally complex problems challenging traditional solvers. Physical implementations of Ising machines include CMOS-based annealers [2]–[5], quantum annealers [6]–[9], photonic systems [10]–[12], memristor networks [13] and more. The underlying **Ising model** was originally introduced to describe the magnetic behavior of spins in ferromagnetic materials [14], [15]. The Ising Hamiltonian (energy) is defined as

$$H(s) = \sum_{i < j} J_{ij} s_i s_j + \sum_i h_i s_i, \quad (1)$$

where spin variables $s_i \in \{-1, +1\}$.

Boolean Satisfiability (SAT) is a decision-making problem seeking an assignment of Boolean variables that makes a given formula true. It was the very first problem proven to be NP-complete [16], [17]. Many other computationally hard problems can be reduced to SAT [18], [19]. SAT has wide-ranging applications, spanning verification [20]–[22], bioinformatics [23], and cryptography [24]–[27], where encryption algorithms encoded into SAT instances enable key-recovery attacks. SAT is a special case of constraint satisfaction problems (CSPs) [28]. **MaxSAT** is the optimization version, where the objective is to find an assignment that satisfies the *maximum number* of clauses (as opposed to *all* under SAT) [29]. MaxSAT is useful when full satisfiability is not possible, and partial solutions are still meaningful [30]. Unlike SAT, a decision problem, MaxSAT as an optimization problem belongs to the class of combinatorial optimization problems (COPs).

Conjunctive Normal Form (CNF) is a standard representation for Boolean formulas representing SAT problems. CNF is structured by *ANDing clauses*; and clauses, by *ORing literals*. Each literal is the equivalent of a variable or its negation [31]. k SAT CNF instances are characterized by the total number of clauses m , the total number of variables by n , and the number of literals per clause (i.e., clause width) k . **Variable Interaction Graphs (VIG)** [32] visualize SAT problems for structural analysis, where variables become nodes, and edges indicate co-appearance of variables in the same clause.

Random SAT problems are uniformly randomly generated SAT instances. Their hardness is strongly related to the clause-to-variable ratio (m/n) [33]. Random problems are typically hardest for a specific m/n , demarcating the phase transition region [34]. These are artificially generated problems to benchmark solvers, especially in the transition region. **Structured SAT instances**, in contrast, mostly stem from real-life problems and exhibit non-uniform variable interactions and modularity that distinguish them from random formulas [35], [36].

Semiprime factorization problems are especially important structured SAT instances because of their role in RSA cryptosystems [37]. A semiprime is a composite of two primes roughly equal in size. This is the hardest form of factorization. The semiprime factorization problem can be expressed as SAT by converting the multiplication circuit (to take the product of prime factors) into CNF [38]. The security of RSA, one of the most widely deployed cryptosystems, relies on the practical hardness of this problem. No efficient algorithm is known for factoring large semiprimes [39]. The best classical methods scale super-polynomially [40]. Any emerging technology that achieves efficient, scalable semiprime factorization can render RSA obsolete by undermining its security [41].

Backbone variables are variables that take the same value in all satisfying assignments, often reflecting overconstrained parts of the problem [42]. **Backdoor variables** are sets of variables whose assignment reduces the remaining instance to

a polynomial-time solvable class [43]. Detecting backbone or backdoor variables can be even harder than solving the SAT instance itself [44]. While traditional solvers can sometimes detect and exploit such variables, Ising-based solvers are generally unaware of them.

SAT solvers are highly optimized classical solvers that have been developed for decades [45]. They fall into two categories: complete solvers and incomplete solvers [46]. Complete solvers like CDCL (Conflict-driven Clause Learning) either solve the problem or prove unsatisfiability [47]. Incomplete solvers may provide quick solutions to large instances, but cannot prove unsatisfiability [48]. Ising machines are similar to incomplete SAT solvers, because of their probabilistic nature and incapacity to prove unsatisfiability.

The Quadratic Unconstrained Binary Optimization (QUBO) model represents combinatorial problems with binary variables $x \in \{0, 1\}^n$ and a quadratic objective

$$\min_{x \in \{0, 1\}^n} x^\top Q x, \quad (2)$$

where Q is an $n \times n$ real matrix capturing variable interactions [49]. QUBO is isomorphic to the Ising Hamiltonian via $x_i = (1 + s_i)/2$ [50], making it a standard target for Ising machines. **SAT to QUBO/Ising Formulation** converts SAT problem constraints into quadratic optimization tasks. In the case of 3SAT, each clause requires at least one additional ancillary variable depending on the formulation. With the formulation proposed in [51], for a 3SAT instance with n variables and m clauses, the resulting QUBO/Ising formulation requires $n + m$ spins. No ancillary variables are needed for 1SAT or 2SAT clauses.

Decomposition refers to breaking down large problems into smaller subproblems that fit the target Ising machine [52]. The number of spins in an Ising machine is limited by construction. This limitation is more pronounced in modern Ising machines as an immature technology. Given the modest scale of existing Ising machines compared to real-world problem sizes, decomposition techniques are essential [53].

Tabu search is a local search method designed for solving COPs [54]. It can be specialized to target specific problem classes, such as SAT within CSPs [55] or QUBO within COPs [56], making it relevant for both classical and QUBO/Ising-based solvers. In this work, we use *D-Wave Tabu* [57] as an idealized QUBO/Ising solver, independent of the hardware limitations of our physical Ising chips.

In classical SAT problems, all clauses, i.e., constraints, are *hard* in the sense that all must be satisfied – unlike MaxSAT where some constraints, referred to as *soft* constraints, may be violated. However, not all hard constraints restrict the search space equally. To make a distinction, we introduce new terminology: **Tight constraints** are highly restrictive constraints (e.g., as induced by backbone variables). **Loose constraints**, by contrast, are less restrictive constraints (clauses) that can prune only smaller portions of the search space. This distinction is especially relevant for Ising machines: While CDCL-style SAT solvers can exploit tight constraints to accelerate

search through propagation and learning, Ising machines often struggle with them as steep energy penalty terms.

III. CHALLENGES OF STRUCTURED SAT

While semiprime factorization can be expressed in SAT form, solving such SAT instances on Ising machines comes with distinct challenges. Ising hardware operates on QUBO or Ising models, both of which are unconstrained. Logical constraints are typically introduced as energy rewards and penalties in the objective function. This introduces several complications, especially for problems that have unique or nearly unique solutions, such as semiprime factorization.

One fundamental issue arises from how *tight* constraints are translated into energy reward/penalties. A variable with a fixed value in the CNF, typically enforced by a unit (i.e., 1-wide, 1SAT) clause like (a) or $(\neg a)$ becomes a linear bias term such as $-a$ or $a-1$ in the QUBO formulation. While this by itself can guide the system toward the global minimum, terms induced by wider clauses can suppress the impact. For instance, a 2SAT clause can introduce QUBO coefficients $-a-b+ab$, which can annul the impact of unit clauses involving variables a or b . By suppressing the influence of unit clauses in this manner, wider clauses can cause the solver to get stuck at local minima.

This is particularly problematic for *backbone variables* which take the same value in all satisfying assignments. SAT solvers may benefit from identifying and exploiting backbones to prune the search space. In contrast, Ising machines suffer, as backbone-heavy problems tend to induce a larger number of tightly constrained variables which leads to poor convergence. We will experimentally validate in Section VII that Ising machine performance degrades as backbone density increases.

Another key limitation of Ising machines lies in their stochastic local search (SLS) nature, which prevents them from detecting logical conflicts. Essentially, Ising solvers operate by flipping bundles of variables in parallel, attempting to minimize the overall energy. However, this process does not guarantee convergence to the global minimum, nor does it provide feedback about unsatisfiable states. In contrast to CDCL-based SAT solvers, which incorporate clause learning and backtracking to avoid invalid regions of the search space, Ising machines lack mechanisms to recognize and recover from conflicts. As a result, they may repeatedly explore invalid or cyclic configurations, ultimately degrading solution quality and convergence.

Finally, size limitations of contemporary Ising machines significantly limit problem sizes for semiprime factorization. Larger semiprime numbers incur a larger number of logic gates in the respective multiplication circuit. The number of extra variables in the CNF grows quadratically in turn. This makes decomposition into smaller subproblems a necessity. In forming a subproblem, some variables are selected to be included, others left out. Decomposition has the side effect of assigning fixed (typically best known) values to left-out variables. Each such fixed assignment introduces yet another *tight* constraint.

Our solution is the hybrid flow from Figure 1, which has two major components: A *CNF preprocessing* step (Section IV) that removes easily detectable *tight* constraints from the CNF; and a systematic *subproblem selection* (i.e., decomposition) mechanism (Section V).

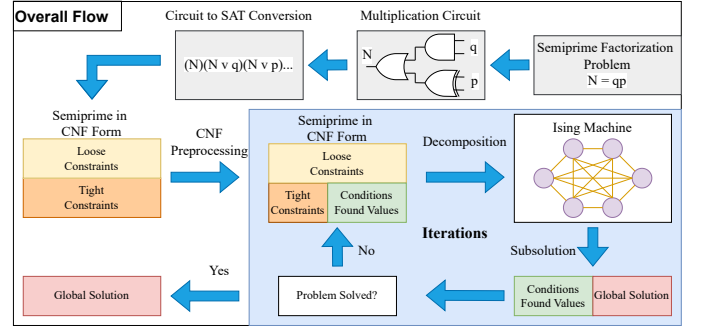


Fig. 1: Proposed Hybrid Flow

IV. CNF PREPROCESSING FOR ISING MACHINES

The first component of our proposed hybrid approach comprises multiple CNF preprocessing techniques to eliminate easily detectable *tight* constraints from the problem. Once the CNF is processed, the remaining part is iteratively decomposed and solved using the Ising machine.

CNF preprocessing is widely used in traditional SAT solvers, however, adaptation to Ising machines requires several modifications. We propose a series of complementary methods, described in the order of application.

A. Gate Encodings for CNF

Each gate in the multiplication circuit used for a semiprime factorization problem is encoded as multiple clauses in the CNF. Some CNF encodings are more suitable for Ising machines due to their clause structure and the lower number of ancillary variables.

To encode an *OR* gate in the CNF, we start with

$$f = c \leftrightarrow (a \vee b)$$

where c is the output of the *OR* gate; a and b , the inputs. The truth table for this relation is given by:

a	b	c	$a \vee b$	$f = c \leftrightarrow (a \vee b)$
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Not all input combinations are valid for an *OR* gate. The combinations that violate the expected behavior must be excluded. In the CNF, this is achieved by adding a constraint, i.e., clause, for each invalid case:

- $(a \vee b \vee \neg c)$ excludes $(a = 0, b = 0, c = 1)$
- $(a \vee \neg b \vee c)$ excludes $(a = 0, b = 1, c = 0)$
- $(\neg a \vee b \vee c)$ excludes $(a = 1, b = 0, c = 0)$
- $(\neg a \vee \neg b \vee c)$ excludes $(a = 1, b = 1, c = 0)$

The complete encoding (*Option-1*) then becomes:

$$(a \vee b \vee \neg c)(a \vee \neg b \vee c)(\neg a \vee b \vee c)(\neg a \vee \neg b \vee c)$$

Option-1 has 4 3SAT clauses. The equivalent Ising/QUBO representation requires 7 physical spins, including 4 ancillaries. Tseitin encoding [58] offers a more compact alternative:

$$c \leftrightarrow (a \vee b)$$

Instead of excluding each invalid case explicitly, Tseitin encoding uses logical implications to enforce the expected gate behavior:

$$\begin{aligned} (a \vee b \vee \neg c) & \text{ excludes } (a = 0, b = 0, c = 1) \\ (\neg a \vee c) & \text{ enforces } a = 1 \Rightarrow c = 1 \\ (\neg b \vee c) & \text{ enforces } b = 1 \Rightarrow c = 1 \end{aligned}$$

This produces *Option-2*:

$$(\neg c \vee a \vee b)(\neg a \vee c)(\neg b \vee c)$$

The QUBO/Ising equivalent of Option-2 requires only 4 physical spins, with a single ancillary variable introduced by the lone 3SAT clause. Using Option-2 instead of Option-1 reduces the physical spin equivalent of each gate by 57%, allowing up to 43% more gates to be mapped to the same Ising hardware spin budget.

Implementation: If the CNF is initially in Option-1 form, gates can be detected and converted into the more spin-efficient Option-2 encodings. Detection is done by grouping clauses that share the same set of variables and then analyzing their literal polarities, to identify gate-specific negative literal count *signatures*. For example, a signature of (1, 1, 1, 2) indicates three clauses with one negative literal and one clause with two, which typically corresponds to an *OR* gate in Option-1 form. Once identified, the respective four clauses are removed and replaced with the three-clause Option-2 encoding, reducing the per-gate spin requirement from 7 to 4. *AND*, *OR*, *NAND*, and *NOR* gates can be encoded with both CNF options. However, *XOR*, *XNOR* gates, and multiplexers are natively expressed in Option-1 form, and Option-2 is not possible.

B. 1SAT Propagation

When semiprime factorization problems are represented as multiplication circuits in the CNF, the circuit output corresponds to the semiprime number to be factorized. These known output values appear in the CNF as unit (1SAT) clauses, hence, translate into *tight* constraints. Such known outputs represent the most obvious backbone variables. While they can be detected and removed easily with traditional methods, as noted earlier, they become challenging once mapped into QUBO form. Although the goal is to solve the whole problem for the unknown input variables on the Ising machine, the fixed output bits can be exploited to prune the CNF before performing the QUBO transformation.

Implementation: 1SAT clauses can be identified by a simple clause-length check and removed from the problem with

unit propagation [59], a lightweight deterministic traditional method: As the name implies, for a positive literal like (*a*) the variable is set to 1; for a negative literal like ($\neg a$), to 0. These assignments are recorded and propagated through the CNF formula. During propagation, satisfied clauses are eliminated, and partially satisfied clauses are reduced.

This process may recursively generate new 1SAT clauses. For example, if the output of an *OR* gate is fixed to 0, both of its input variables must also be 0, resulting in two additional unit clauses – which can be handled in the same manner. Additionally, some clauses are reduced from 3SAT to 2SAT, which improves compatibility with QUBO formulation because 2SAT clauses do not require ancillary spins. This simplification reduces both the variable count and clause complexity, improving the quality and scalability of the resulting QUBO instance.

C. 2SAT Conditioning

Single-input gates, such as *NOT* and *BUFFER*, can be represented in the CNF as pairs of 2-literal (2SAT) clauses. These gates may already exist in the original circuit encoding, although this is relatively rare, or they may emerge naturally as a by-product of other simplifications—most commonly after 1SAT propagation. 2SAT conditioning is our most effective CNF preprocessing method, serving as the foundation for many of the other preprocessing methods we will present in this section.

A *NOT* gate, for example, can be expressed in CNF as:

$$(a \vee b)(\neg a \vee \neg b)$$

and a *BUFFER* gate as:

$$(a \vee \neg b)(\neg a \vee b).$$

Semiprime multiplication circuits seldomly contain such gates in their native form. These gates frequently appear when more complex gates are simplified. For example, an *XOR* gate is typically encoded using four 3SAT clauses as such:

$$(a \vee b \vee \neg c)(\neg a \vee b \vee c)(a \vee \neg b \vee c)(\neg a \vee \neg b \vee \neg c).$$

If the output of the *XOR* gate is known – say $c = 1$ – the CNF collapses to:

$$(a \vee b)(\neg a \vee \neg b),$$

which is precisely the CNF of a *NOT* gate.

Both *BUFFER* and *NOT* gates enforce a *tight* constraint between two variables: *BUFFER* enforces $a = b$; *NOT*, $a = \neg b$. This makes them functionally equivalent to backbones: once one variable is fixed, the other is fully determined. Such *tight* constraints degrade Ising machine performance if not removed. In SAT solving, such constraints reduce the search space and often accelerate convergence. In QUBO form, however, they are problematic for Ising solvers. The constraint is typically encoded as an energy penalty term. Such penalty terms can be outweighed by competing energy reward/penalty terms, leading the solver to return logically invalid assignments.

In traditional CDCL-based SAT solvers, 2SAT pairs present a specific property: they form *branching dead ends*. Because both clauses are logically equivalent to a deterministic relationship between the respective two variables, they cannot produce new conflicts during the search. Modern SAT solvers exploit this by applying *Blocked Clause Elimination* (BCE), which removes one of the two clauses during CNF preprocessing while preserving satisfiability [60]. The simplification is safe for the traditional solvers and accelerates them.

Ising-based QUBO solvers are fundamentally different. They are *conflict-unaware*: There is no clause learning, no branching based on logical conflicts, and no way to take advantage of BCE. The logical equivalence between two variables is implemented as a weighted penalty term in the energy function, and there is no guarantee that the solver will satisfy it if violating it produces a lower-energy (but logically invalid) state.

2SAT Conditioning is designated to address this. Rather than keeping one or both clauses in the CNF, we remove both and explicitly record the deterministic constraint between the respective variables. For the *NOT* gate $(a \vee b)(\neg a \vee \neg b)$, we remove both clauses, and store the hard condition $a = \neg b$. One variable is designated as the *master variable* – it remains in the CNF – and the other becomes a *replaced variable*, whose value will always be derived from the master according to the recorded condition.

The recorded condition list is used after each Ising solver iteration to reconstruct the values of replaced variables from their masters, ensuring logical consistency in the global solution. When applied systematically, 2SAT conditioning can effectively eliminate *tight* constraints.

Implementation: In detecting 2SAT conditions, as well, we group clauses by the set of variables they contain to derive negative literal count signatures, following the very same procedure from Section IV-A. Matching these signatures or encoded patterns against a predefined lookup table allows us to identify gates. For example, the signature (0, 2) corresponds to a *NOT* gate; and (1, 1), to a *BUFFER* gate.

Occasionally, three distinct 2SAT clauses over the same two variables appear with different signatures, such as (0, 1, 2), (0, 1, 1), or (1, 1, 2). Since two variables have exactly 4 possible truth assignments, the presence of three distinct 2SAT clauses constraints three of them, leaving only one satisfying assignment. For example, the set

$$(a \vee b), \quad (a \vee \neg b), \quad (\neg a \vee b)$$

has the signature (0, 1, 1), and the only unconstrained assignment is $(\neg a \vee \neg b)$, meaning both a and b must be 1 to satisfy all 3 clauses. In such cases, we can determine the values of both variables outright, and add the equivalent 1SAT such as (a) and (b) to the CNF to enforce the assignments. Such 1SAT clauses are then handled using standard unit propagation.

Although the idea of replacing equivalent variables is conceptually simple, the implementation becomes challenging in practice due to *chained conditions*. A naïve approach would be to traverse the CNF immediately after detecting each

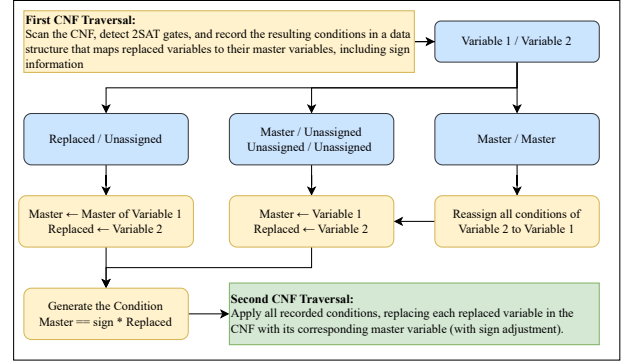


Fig. 2: 2SAT Conditioning Decision Flow

condition, replacing one variable with another throughout. If another condition is later found, the CNF must be traversed again. This leads to excessive passes over the CNF and poor performance.

Chained conditions pose the more subtle issue. Suppose that we detect the condition $a=b$ and decide to keep the variable b in the CNF while replacing a . During traversal, we detect another condition $b=c$, and decide to keep the variable c while replacing b . At the end of preprocessing, both a and b would be gone from the CNF, and only c would remain – but without careful bookkeeping, we can lose or inconsistently apply the original $a=b$ relation.

To solve both inefficiency and chaining problems, we designed a *two-pass conditioning algorithm* that completes all processing in exactly two CNF traversals as shown in Figure 2. To safely manage chaining, we track three disjoint sets of variables: *masters* (kept in the CNF), *replaced* (substituted later by their master), and *unassigned*. When a 2SAT gate is encountered, the classification of its two variables determines the action as indicated in Figure 2:

- Both unassigned: One becomes master, the other replaced.
- One master, one unassigned: The unassigned is replaced.
- One replaced, one unassigned: The unassigned inherits the master (with sign preserved).
- Both masters: One is promoted to *grand master*, references updated accordingly.

After traversal, replaced variables are substituted by their masters, removing *tight* conditions before QUBO conversion. The condition list is retained to reconstruct full-fledged assignments after Ising iterations.

D. Replaced Value Propagation:

When **2SAT conditioning** is applied, it produces a *condition list* mapping replaced variables to their master variables and the corresponding sign relation. If **1SAT propagation** is performed after 2SAT conditioning, it may assign values to some variables that also appear in the condition list. In such cases, the assigned value can be immediately propagated to the replaced variable without revisiting the CNF. For example, if 2SAT conditioning recorded $a=b$ and 1SAT propagation

assigns $b=1$, then we can directly deduce $a=1$ as seen in Figure 3. Similarly, if $a=\neg b$ and $b=0$, then $a=1$.

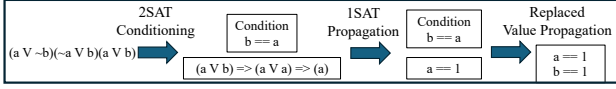


Fig. 3: Replaced Value Propagation

This process can cascade: a newly assigned replaced variable may itself be the master of other replaced variables in the condition list, allowing further assignments.

This method only applies when *2SAT conditioning is followed by 1SAT propagation*. It exploits the synergy between the two steps to assign additional variable values without touching the CNF clauses again, and reduces the condition list length. If a variable's value can be determined deterministically through simple preprocessing (e.g., 1SAT propagation, 2SAT conditioning, or replaced value propagation), it is highly constrained and effectively acts as a backbone. Such variables have no freedom in the solution space, making them unnecessary to send to Ising hardware that may return approximate solutions. Removing them before QUBO conversion both saves spins and prevents additional *tight* constraints that can hinder the Ising machine performance.

E. Clause Cleaning

Good CNF preprocessing methods are simple and complementary, and their strength becomes apparent when one technique triggers another [47]. In our approach, 1SAT propagation initiates 2SAT conditioning, which in turn enables replaced value propagation and clause cleaning. 2SAT conditioning often replaces variables with their master equivalents. This can unintentionally create duplicate literals or tautological clauses, both of which add no useful constraints. Clause cleaning is a lightweight step to remove these redundancies before QUBO conversion. An example is given in Figure 4.

If a replaced variable is substituted by its master and that master literal already appears in the clause:

- With the *same sign*, the duplicate can be removed, reducing clause width and the ancillary count.
- With the *opposite sign*, the clause becomes a tautology and is always satisfied, so it can be removed entirely.

Reducing literal count not only shrinks the CNF size but also relaxes the ancillary spin requirements for clause-to-QUBO mapping, enabling more problem variables (rather than ancillaries) to be embedded on spin-limited Ising hardware.

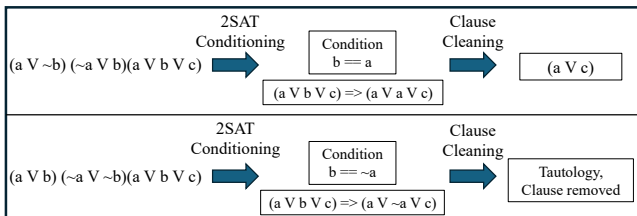


Fig. 4: Clause Cleaning

This approach avoids additional CNF passes while producing a cleaner, smaller problem that is more Ising hardware-friendly.

F. Traditional Methods

We integrate several well-known preprocessing methods into our flow to evaluate their impact.

1) *Subsumption Reduction*:: After earlier simplifications such as 1SAT propagation or 2SAT conditioning, some clauses become subsumed by smaller ones and can be removed. This is especially useful when a 3SAT clause is subsumed by a 2SAT clause, since the QUBO equivalent then requires fewer ancillary spins, reducing hardware demand. A side effect, however, is that it alters the variable interaction graph (VIG) which may complicate decomposition – we will discuss this effect in Section VII.

2) *Pure Literal Elimination*:: Pure literal elimination is one of the earliest SAT preprocessing techniques [45]. If a variable appears with only one polarity in the CNF, it can be assigned consistently and then removed from the formula.

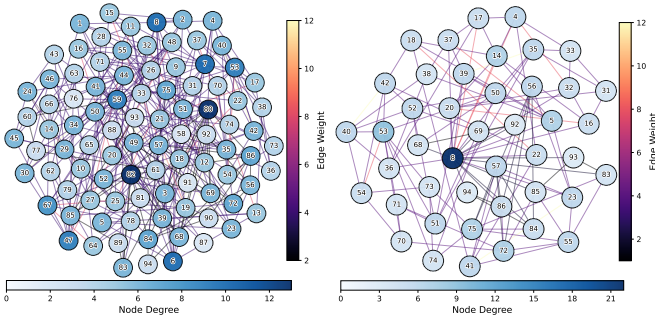
3) *Tightly-Constrained Variable Branching*:: Applying 2SAT conditioning together with other simplification methods can collapse many variables into a single master variable. This consolidation increases the master variable's connectivity in the VIG, and results in larger QUBO/Ising interaction coefficients, often exceeding the target Ising machine's coefficient limits. In addition, such highly connected variables may complicate problem decomposition.

In the SAT solving context, such variables represent ideal branching candidates for conflict detection. Our goal, however, is different. We are after reducing the problem size and removing tightly constrained variables. Inspired by SAT solvers, we perform a *single branching step* on each such variable with a random assignment. This effectively prunes the search space: In roughly half of the solver runs, the branch immediately closes (no solution), while in the other half the problem continues with a substantially smaller variable set. Figure 5a illustrates this effect: After simplifications (Figure 5b), variable 8's node degree rises from 12 to 21, making it the dominant tightly-constrained variable. Branching on it immediately shrinks the problem for half of the repeats, often accelerating convergence.

V. SAT DECOMPOSITION

The role of preprocessing is to make the CNF as “Ising-friendly” as possible – by removing tight constraints, reducing variable count, and simplifying clause structure – before it is passed to the hardware. Even after extensive CNF preprocessing, the resulting problem may still exceed the spin limit of the target Ising hardware. In the end, Ising machine capacity cannot keep up with increasing problem sizes of practical importance. *Problem decomposition* – selecting and solving subproblems small enough to fit on the Ising hardware, then iteratively combining partial results – therefore is almost always inevitable for real-life problems.

Beyond hardware feasibility, decomposition directly affects performance. As the gap between typical problem sizes and the



(a) Before preprocessing
($n/m = 94/352$, gates = 95)

(b) After preprocessing
($n/m = 44/145$, gates = 22)

Fig. 5: Impact of *tightly-constrained variable branching* considering an 8-bit semiprime factorization example.

Ising machine capacity widens, careful subproblem selection becomes increasingly critical. We adapt existing subproblem selection approaches to the structured nature of semiprime factorization CNFs and propose new heuristics to improve convergence further.

A. BFS vs. DFS based Subproblem Selection

For random kSAT instances, subproblem selection on the VIG via breadth-first search (BFS) is effective [1], as it selects compact, highly connected local clusters that propagate assignments efficiently. However, structured CNFs, such as those from semiprime factorization circuits, behave differently.

In semiprime factorization, the outputs are known, and the task is to recover the inputs. The VIG is highly clustered: Each logic gate’s variables form dense subgraphs. BFS tends to select all variables of a single gate (and possibly its immediate neighbors) into the subproblem, while freezing fan-in and fan-out variables by assigning their last known value in the global solution to the CNF. This effectively forces the gate into a fixed state locally. If the frozen values are incorrect – which is common early in the search – this locks in wrong assignments and slows global convergence.

Depth-first search (DFS) offers a better solution. By following long paths through the VIG, DFS selects variables along chains of gates, often including outputs with only partial input coverage. This allows the Ising solver to optimize multiple gates in a sequence, propagating constraints along the input–output chain and accelerating convergence. Empirically, DFS avoids the “local trap” effect of BFS and better aligns with the directional flow of information in these circuits. As illustrated in Figure 6, BFS groups variables inside a single gate such as *AND2*, excluding its fan-in and fan-out, which can bias the local assignment toward incorrect values such as $a=0$ and $c=0$. In contrast, DFS is more likely to select a consistent input–output chain such as $(a=1, b=1, c=1)$, which matches the correct solution for this slice.

B. Variable Filtering

Structured problems often contain a few tightly-constrained variables. It happens especially after CNF preprocessing steps like 2SAT conditioning that merge many variables into a

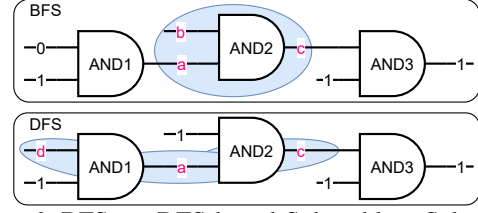


Fig. 6: BFS vs. DFS based Subproblem Selection

single master, like variable 8 in Figure 5b. These high-degree variables get usually repeatedly selected by graph-traversal based decomposition (be it BFS or DFS), causing their assignments to oscillate as the Ising solver optimizes different subsets. To mitigate this, we introduce a *variable filtering* heuristic: if a variable appears in the subproblem selection for x consecutive iterations (e.g., $x=5$), its selection priority is temporarily reduced for the next x iterations. This allows the effects of its current value to propagate to other variables before it is reconsidered and prevents oscillations, improving convergence stability.

C. No In-Loop Preprocessing

CNF preprocessing methods are applied together before decomposition begins, and are not useful thereafter. During decomposition, the selection process continually freezes variables to their current best-known values – many of which are wrong until a global solution is found. This transforms the SAT instance to a MaxSAT instance, which is then converted to QUBO form. The Ising machine therefore tries to maximize the number of satisfied clauses rather than satisfy all of them.

In MaxSAT mode, further CNF simplification distorts the clause balance, and thereby, QUBO energy reward/penalty terms. For example, a CNF such as $(a \vee b)(a \vee \neg b)(\neg a \vee b)(a \vee \neg b \vee c)$ has a single solution where $a=1, b=1$. If the variable a is frozen incorrectly to 0 during the decomposition step, the CNF becomes $(b)(\neg b)(\neg b \vee c)$ which is a MaxSAT instance due to the presence of conflicting unit clauses (b) and $(\neg b)$. The Ising machine will try to satisfy as many clauses as possible, producing $b=0$ or $b=1$ depending on the energy dynamics. If we then simplify $(\neg b \vee c)$ as subsumed by $(\neg b)$, we distort this balance, making recovery of the correct assignment less likely.

Our experiments confirm that any CNF modification during decomposition almost always makes the problem unsolvable on the Ising machine. We therefore exclude CNF preprocessing during decomposition passes.

VI. EVALUATION SETUP

Ising Testbed: We use a platform with 8 45-spin all-to-all connected CMOS Ising chips. This Ising machine’s coefficient range is $[-14, +14]$. Chips are integrated onto a PCIe board with FPGA-based control, enabling direct communication with a host server. Multiple boards are integrated on a PCIe riser, enabling faster parallel runs [61]. The boards are hosted on a server equipped with an Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz.

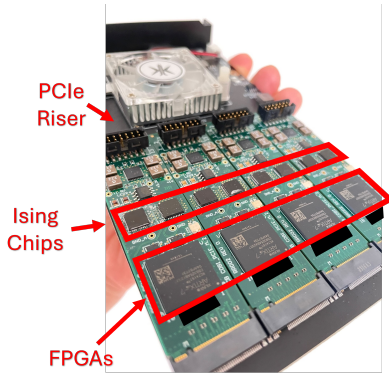


Fig. 7: Ising Chips with the PCIe Riser

Metrics and Terminology: *Instance* is a single SAT problem in CNF form. *Repeat* is an independent run of the Ising solver on the same SAT instance. Multiple repeats account for the stochastic nature of the algorithms and ensure statistical significance. *Iteration* is one complete cycle of decomposition, embedding, and solver call. *Time to Solution (TTS)* is the expected iteration count required to solve an instance with a 95% success probability, computed from the per-run success probability [62].

Benchmarks: We use two types of benchmarks:

- *SATLIB controlled-backbone* [63]: Publicly available 3SAT instances with specified backbone sizes. These enable systematic variation in solution-space structure and provide a complementary test for decomposition methods.
- *Semiprime factorization* [64]: 79 CNF instances derived from schoolbook multiplier circuits with known outputs and unknown inputs, representing the structured problems targeted by our preprocessing and decomposition flow. Instance properties are summarized in Table I and Table II.

Controlled-backbone instances serve as an intermediate benchmark between highly structured problems and fully random formulas. Semiprime factorization represents a strongly structured class of problems.

TABLE I: CNF encoding parameters for semiprime instances grouped by output precision (#Bits)

# Bits	#Instances	CNF Parameters		
		k	n	m
4	1	3	21	72
5	3	3	37	133
7	4	3	76	283
8	11	3	94	352
10	20	3	162	618
11	40	3	190	727

Classical Baseline: We use *D-Wave Tabu* solver [57], which implements a Tabu Search metaheuristic optimized for QUBO formulations [56]. While running on classical hardware, it is representative of the QUBO formulation style, coefficient handling, and local search behavior expected from D-Wave’s quantum annealers, making it a suitable baseline for evaluating QUBO-based problem-solving performance. The timeout for

TABLE II: Gate counts for semiprime instances grouped by output precision (#Bits).

#Bits	Gate Counts				
	AND	OR	XOR	XNOR	Total
4	9	4	4	0	17
5	14	9	7	4	34
7	29	15	20	10	74
8	35	16	26	18	95
10	64	28	49	22	163
11	75	33	58	26	192

the solver is set to 5ms, and the subproblem size is fixed as 45, the size of the Ising chips. Tabu solver runs are executed on an HPC cluster of AMD EPYC 7763 64-core processors.

VII. EVALUATION

We evaluate our flow using four complementary studies: First, we analyze solver behavior on controlled-backbone random SAT instances, which serve as synthetic benchmarks for understanding how tightly-constrained variables impact Ising-based methods. This baseline, largely absent from prior Ising literature, provides a bridge between random and structured SAT problems. Next, we measure the effects of CNF preprocessing on semiprime factorization encodings. We then report the resulting TTS distributions under different preprocessing and decomposition strategies. Finally, we analyze end-to-end runtime tradeoffs, balancing preprocessing overhead against solver performance. Together, these results highlight both the challenges and opportunities of structured SAT solving on Ising machines.

A. Backbone Analysis

To establish a controlled baseline, we first study Ising solver behavior on SATLIB’s Controlled-Backbone-Size (CBS) random 3-SAT benchmarks. These synthetic instances allow us to vary backbone fraction and clause density in a principled way, thereby simulating the implications of tight constraints.

We use SATLIB’s Controlled-Backbone-Size (CBS) random 3-SAT instances with $n=100$ variables and all available clause counts $m \in \{403, 411, 418, 423, 429, 435, 441, 449\}$ as well as backbone fractions $b \in \{10, 30, 50, 70, 90\}\%$ as a *controlled case study* to vary tight-constraint pressure. For each (m, b) bucket we take the first 100 instances and run 100 independent repeats of the same preprocessing and decomposition flow on (i) the D-Wave Tabu solver and (ii) our Ising chips, with a per-repeat cap of 5000 iterations. All problem instances have at least one successful repeat under all conditions in the given iteration limit. In our runs, every instance achieved at least one successful repeat within the 5,000-iteration cap, but the success rates vary.

Figure 8 reveals that, as the backbone fraction b increases (10% $\rightarrow$ 90%), successful repeats drop for all clause counts m . The change in the curves is close to piece-wise linear, with similar slopes and simple vertical offsets by m . Overall, larger backbones increase the impact of tight-constraints, making the problem harder for Ising solvers. In these graphs, the backbone effect appears more pronounced because decomposition

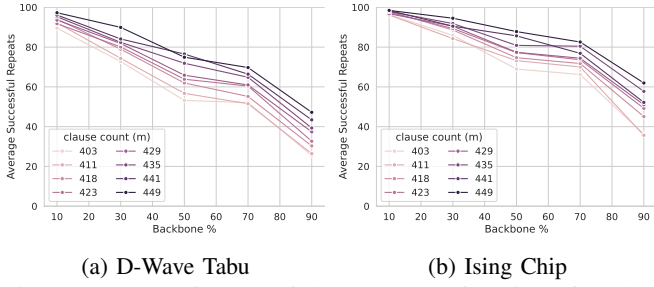


Fig. 8: Number of successful repeats as a function of % share of backbone variables in the problem.

amplifies this pressure. The smallest CBS instance, with 100 variables and 403 clauses, is transformed into 503 QUBO variables and solved on a 45-spin all-to-all connected Ising chip – about 11 times smaller than the original problem size – allowing the impact of tight constraints to be observed.

Interestingly, backbone behavior shows overlap between Stochastic Local Search (SLS) SAT solvers and Ising solvers. For SLS solvers, the hardest cases often arise when formulas contain fewer clauses but large backbones [65]. We observe the same trend for Ising solvers, despite fewer clauses implying fewer ancillary variables and higher utilization of Ising solver capacity. Understanding this connection remains an open question for future work.

B. Semiprime Factorization Problems

The central performance metric for our study is *time-to-solution* (TTS), which reflects both solver convergence and the effect of preprocessing on structured instances. Here we evaluate practical solver performance.

We consider 79 semiprime instances which cover a wide range for the number of output bits, tested under varying levels of preprocessing and using two decomposition strategies (BFS vs. DFS) on both the Ising chip and the D-Wave Tabu solver. For each configuration, we record both the TTS distribution and the fraction of instances successfully solved.

Table III summarizes the effect of incremental CNF preprocessing on the average variable count across semiprime factorization instances. As expected, higher preprocessing levels steadily reduce the problem size, in some cases solving the instance outright (indicated by 0 variables remaining).

TABLE III: Average variable count in the semiprime CNF after each level of CNF preprocessing. Each column specifies a different output precision, i.e., a different number of bits to represent circuit outputs, which serves as a proxy for the multiplication circuit complexity.

Preprocessing Level	4 Bits	5 Bits	7 Bits	8 Bits	10 Bits	11 Bits
0 Original	21	37	76	94	162	190
1 +CNF Encodings	21	37	76	94	162	190
2 +ISAT Propagation	16	29	64	81	145	172
3 +2SAT Conditioning	7	7	41	49	90	123
4 +Replaced Value Prop.	0	0	36	45	83	119
5 +Clause Cleaning	0	0	36	45	83	119
6 +Subsumption	0	0	36	45	83	119
7 +Pure Literal	0	0	35	44	82	118
7 +Branching	0	0	8	7	40	73

Figure 9 shows the percentage of solved instances under BFS-based subproblem selection at different preprocessing levels. For small semiprimes (4, 5, 7, and 8 bits), all instances are solvable regardless of decomposition or preprocessing. For larger semiprimes (10 and 11 bits), the effect is more pronounced: BFS on Ising chips outperforms the D-Wave Tabu solver at low preprocessing levels (0–1). However, BFS performance is highly sensitive to changes in the variable interaction graph (VIG) structure introduced by preprocessing. As a result, the number of solved instances fluctuates, even when tightly constrained variables are eliminated.

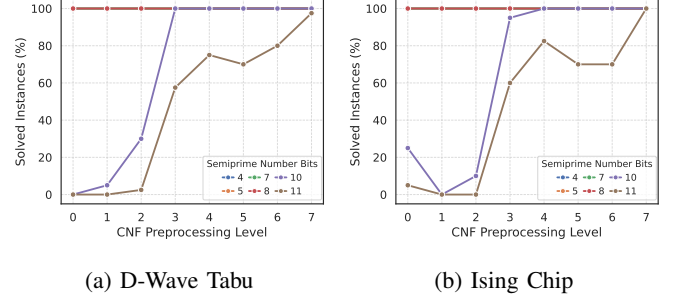


Fig. 9: % solved instances with increasing preprocessing level under BFS-based decomposition.

Figure 10 presents the corresponding results for DFS-based subproblem selection. DFS proves more effective for semiprime factorization, with the Ising chips showing consistent improvements at each preprocessing step. Importantly, once all preprocessing steps are applied, the Ising chip is able to solve all 40 instances of 11-bit semiprime factorization, outperforming the D-Wave Tabu solver in both stability and scalability.

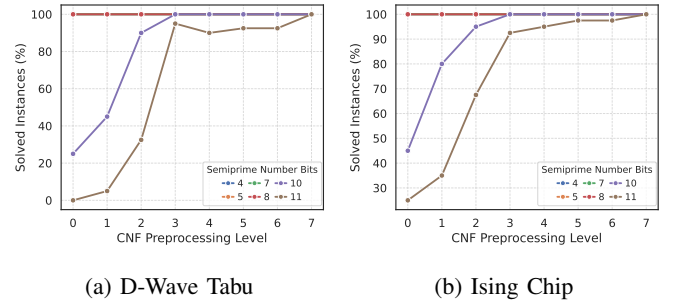
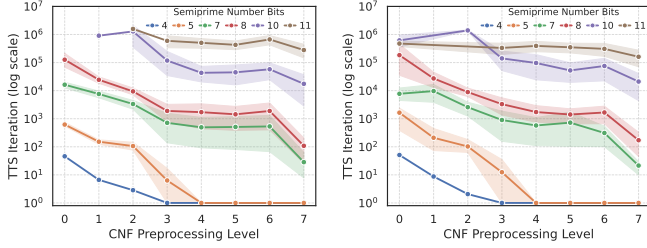


Fig. 10: % solved instances with increasing preprocessing level under DFS-based decomposition.

When all instances are solvable (e.g., 4–8 bit semiprimes), time-to-solution (TTS) becomes the main metric for comparison. Figure 11 reports TTS under BFS selection. Consistent with solved instance percentages, BFS performance fluctuates with preprocessing due to its sensitivity to the VIG structure. Even for fully solvable cases, TTS varies significantly across preprocessing levels.

Figure 12 shows TTS for DFS-based decomposition. Here, semiprimes up to 10 bits exhibit stable performance because

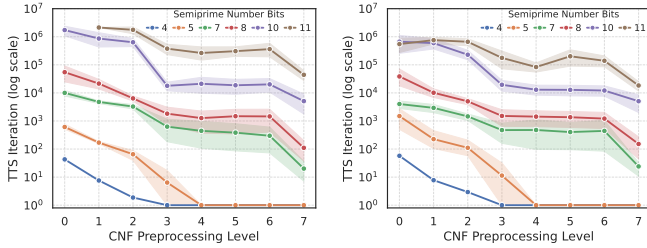


(a) D-Wave Tabu

(b) Ising Chip

Fig. 11: TTS as a function of CNF preprocessing level under BFS-based decomposition.

DFS is less sensitive to preprocessing. For 11-bit semiprimers, fluctuations appear as more instances become solvable and visible in TTS calculation. Overall, the Ising chip consistently yields lower TTS values than D-Wave Tabu, reflecting higher-quality subproblem solutions.



(a) D-Wave Tabu

(b) Ising Chip

Fig. 12: TTS as a function of CNF preprocessing level under DFS-based decomposition.

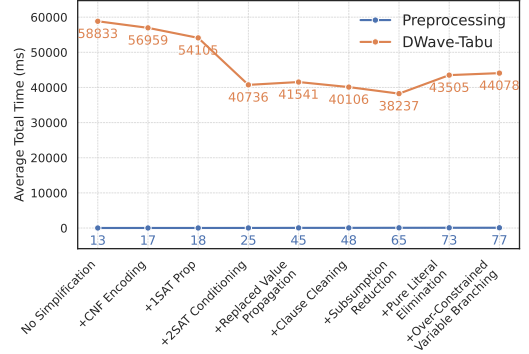
In summary, these results demonstrate that reducing adverse effects of tightly constrained variables through preprocessing improves the capability of Ising machines, accelerates convergence, and reduces TTS. Interestingly, variable count alone does not determine performance: for example, 4- and 5-bit semiprimers at level 3 both reduce to 7 variables (directly embeddable on hardware), yet their TTS values in Figures 11b and 12b differ by an order of magnitude. This highlights the critical role of constraint tightness and CNF structure. With preprocessing, the effective reach of the Ising machine extends from 8-bit semiprimers (94 variables, solvable without preprocessing) to 11-bit semiprimers (190 variables), doubling the scale of tractable problems.

C. Runtime Analysis

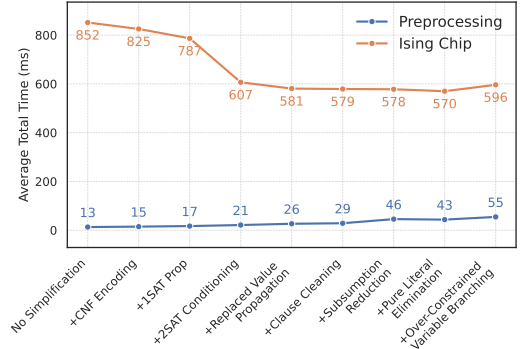
While TTS distributions capture solver convergence behavior, practical usability also depends on end-to-end runtime. Preprocessing introduces additional CPU overhead, but can dramatically reduce solver effort on the Ising chip. In this subsection, we analyze this tradeoff, comparing preprocessing time against chip time across the entire problem set.

Our implementations are written in Python 3.8.10, and preprocessing includes CNF parsing, graph generation, and simplification passes. Average preprocessing time under the no-simplification setting is about 13 ms, but subsequent passes

add varying amounts of overhead. We report how these costs accumulate, and how they compare to the reduction in chip iterations for both the Ising hardware and the D-Wave Tabu solver.



(a) D-Wave Tabu



(b) Ising Chip

Fig. 13: Preprocessing vs. solver Time.

The chip time is computed as the product of the chip annealing time and the number of hardware calls. A substantial reduction in chip time is observed with minimal preprocessing, up to the 2SAT conditioning step. Beyond this point, methods up to clause cleaning produce only slight improvements in chip time, accompanied by a small increase in preprocessing time. Subsumption reduction and pure literal elimination result in considerably longer preprocessing times with almost no improvement in solver performance. Tightly-constrained variable branching produces the highest preprocessing time, and while it accelerates the solved repeats, it also causes half of the repeats to reach the timeout limit, which increases the total chip time.

VIII. RELATED WORK

Literature on *semiprime factorization in SAT form* mostly focuses on performance comparisons. The authors in [38] quantify the runtime of semiprime factorization represented as CNF, using SAT solvers, quantum annealers, and mathematical methods. This work analyzes multiple semiprime circuit encodings and the scaling behavior with the output bit precision. Another study [66] applies SAT solvers to accelerate a subtask of a mathematical method for semiprime factorization, and concludes that the approach is not sufficiently effective.

Research in *prime factorization on quantum annealers* primarily covers embedding as large problems as possible into connectivity-limited quantum hardware. A recent paper [67] introduces a modular embedding method for larger semiprime factoring problems into D-Wave's (relatively sparse) Pegasus architecture. Another recent study [68] covers alternative mathematical reformulation (HUBO) and optimized embeddings. The work in [69] encodes prime factorization as a higher-order polynomial, then reduces it to QUBO with ancillary variables to be mapped on a quantum annealer. Proposals for accelerating quantum annealers for prime factorization problems by exploiting hardware-level controls such as per-qubit anneal offsets also exist [70]. Finally, a large body of work present decomposers for quantum annealers [1], [52], [61].

These approaches advance either the problem formulation in CNF/QUBO form or the efficiency of annealing. However, the combined challenges of problem structure induced tight constraints in QUBO, the inevitable amplification thereof by decomposition, and the design of QUBO/Ising-friendly preprocessing methods have not, to the best of our knowledge, been systematically covered up to now. This is the main focus and contribution of our work.

IX. CONCLUSION

Our study reveals three fundamental obstacles in solving structured SAT problems on Ising hardware: (i) Single-solution instances overconstrain Ising solvers; (ii) Ising hardware lacks conflict-driven learning, making it poorly suited for tightly constrained problems; and (iii) circuit-based encodings scale quickly beyond hardware limits, requiring problem decomposition that further amplifies constraint pressure.

To address these challenges, we combine CNF preprocessing to relax tight constraints with structure-aware problem decomposition. With a lightweight pipeline, we more than double the solvable problem size on a 45-spin all-to-all Ising chip, from 8-bit (94 variables) to 11-bit (190 variables), without hardware changes. Overall, we demonstrate that hardware-aware problem shaping – through preprocessing and informed subproblem selection – can significantly expand the reach of emerging Ising machines on structured SAT problems, which represents an important step toward solving practical real-world problems at scale.

REFERENCES

- [1] H. Cilasun, Z. Zeng, A. Kumar, H. Lo, W. Cho, W. Moy, C. H. Kim, U. R. Karpuzcu, and S. S. Sapatnekar, "3sat on an all-to-all-connected cmos ising solver chip," *Scientific Reports*, vol. 14, no. 1, p. 10757, 2024.
- [2] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, 2015.
- [3] H. Lo, W. Moy, H. Yu, S. Sapatnekar, and C. H. Kim, "An ising solver chip based on coupled ring oscillators with a 48-node all-to-all connected array architecture," *Nature Electronics*, vol. 6, no. 10, pp. 771–778, 2023.
- [4] H. Cilasun, Z. Zeng, S. Ramprasath, A. Kumar, H. Lo, W. Cho, W. Moy, C. H. Kim, U. R. Karpuzcu, and S. S. Sapatnekar, "3sat on an all-to-all-connected cmos ising solver chip," *Scientific Reports*, vol. 14, no. 1, p. 10757, 2024.
- [5] H. Cilasun, W. Moy, Z. Zeng, T. Islam, H. Lo, A. Vanasse, M. Tan, M. Anees, S. Ramprasath, A. Kumar, S. S. Sapatnekar, C. H. Kim, and U. R. Karpuzcu, "A coupled-oscillator-based ising chip for combinatorial optimization," *Nature Electronics*, vol. 8, no. 6, pp. 537–546, 2025.
- [6] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.
- [7] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, "Next-generation topology of d-wave quantum processors," *arXiv preprint arXiv:2003.00133*, 2020, available at <https://arxiv.org/abs/2003.00133>.
- [8] S. Ebadi, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho *et al.*, "Quantum phases of matter on a 256-atom programmable quantum simulator," *Nature*, vol. 595, no. 7866, pp. 227–232, 2021.
- [9] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye *et al.*, "Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms," *Nature*, vol. 595, no. 7866, pp. 233–238, 2021.
- [10] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Inoue, S. Utsunomiya, and Y. Yamamoto, "A coherent ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016.
- [11] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, and Y. Yamamoto, "A fully programmable 100-spin coherent ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.
- [12] B. Wu, W. Zhang, S. Zhang, H. Zhou, Z. Ruan, M. Li, D. Huang, J. Dong, and X. Zhang, "A monolithically integrated optical ising machine," *Nature Communications*, vol. 16, no. 1, p. 4296, 2025.
- [13] F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, M. Foltin, S. Yu, Q. Xia, J. J. Yang, R. Beausoleil, W. D. Lu, and J. P. Strachan, "Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks," *Nature Electronics*, vol. 3, no. 7, pp. 409–418, 2020.
- [14] W. Lenz, "Beiträge zum verständnis der magnetischen eigenschaften in festen körnern," *Physikalische Zeitschrift*, vol. 21, pp. 613–615, 1920.
- [15] E. Ising, "Beitrag zur theorie des ferromagnetismus," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925.
- [16] S. A. Cook, "The Complexity of Theorem-Proving Procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.
- [17] L. A. Levin, "Universal Sequential Search Problems," *Problems of information transmission*, vol. 9, no. 3, pp. 265–266, 1973.
- [18] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA: Springer US, 1972, pp. 85–103. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9
- [19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*, first edition ed. W. H. Freeman, 1979.
- [20] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking using SAT procedures instead of BDDs," in *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference (DAC)*, 1999, pp. 317–320.
- [21] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Proc. 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, vol. 1579. Springer, 1999, pp. 193–207.
- [22] K. Jia and M. Rinard, "Efficient Exact Verification of Binarized Neural Networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1782–1795, 2020.
- [23] I. Lynce and J. Marques-Silva, "Sat in bioinformatics: Making the case with haplotype inference," in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*. Springer, Berlin, Germany, 2006, pp. 136–141.
- [24] A. Semenov, O. Zaikin, D. Bespalov, and M. Posypkin, "Parallel logical cryptanalysis of the generator a5/1 in bnb-grid system," in *Proceedings of the International Conference on Parallel Computing Technologies*. Springer, Berlin, Germany, 2011, pp. 473–483.

- [25] A. Semenov, O. Zaikin, I. Otpuschennikov, S. Kochemazov, and A. Ignatiev, "On cryptographic attacks using backdoors for SAT," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, vol. 32, no. 1. AAAI Press, Palo Alto, CA, USA, 2018.
- [26] G. Bard, *Algebraic Cryptanalysis*. New York, NY, USA: Springer Science & Business Media, 2009.
- [27] N. T. Courtois and G. V. Bard, "Algebraic cryptanalysis of the data encryption standard," in *Proceedings of the 11th IMA International Conference on Cryptography and Coding*. Springer, Berlin, Germany, 2007, pp. 152–169.
- [28] F. Rossi, P. V. Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier, 2006, vol. 2.
- [29] W. Zhang, "Phase transitions and backbones of 3-SAT and maximum 3-SAT," in *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP 2001)*. Springer, Berlin, Germany, November 2001, pp. –.
- [30] T. Stützle, H. Hoos, and A. Roli, "A review of the literature on local search algorithms for max-SAT," Intellectics Group, Darmstadt University of Technology, Darmstadt, Germany, Rapport Technique AIDA-01-02, 2001.
- [31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022.
- [32] I. Rish and R. Dechter, "Resolution Versus Search: Two Strategies for SAT," *J. Autom. Reason.*, vol. 24, no. 1–2, pp. 225–275, 2000.
- [33] D. Mitchell, B. Selman, and H. Levesque, "Hard and Easy Distributions of SAT Problems," in *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI)*, vol. 1. AAAI Press, 1992, pp. 459–465.
- [34] P. C. Cheeseman, B. Kanefsky, and W. M. Taylor, "Where the Really Hard Problems Are," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 1, 1991, pp. 331–337.
- [35] C. Ansótegui, M. L. Bonet, and J. Levy, "Towards industrial-like random sat instances," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 387–392.
- [36] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, J. Levy, and L. Simon, "Community structure in industrial SAT instances," *Journal of Artificial Intelligence Research*, vol. 66, pp. 443–472, 2019.
- [37] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [38] M. Mosca and S. R. Verschoor, "Factoring semi-primes with (quantum) SAT-solvers," *Scientific Reports*, vol. 12, no. 1, pp. 7982:1–7982:12, 2022.
- [39] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.
- [40] A. K. Lenstra and H. W. Lenstra, Eds., *The Development of the Number Field Sieve*, ser. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, 1993, vol. 1554.
- [41] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. Santa Fe, NM, USA: IEEE, 1994, pp. 124–134.
- [42] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, "Determining computational complexity from characteristic 'phase transitions'," *Nature*, vol. 400, no. 6740, pp. 133–137, 1999.
- [43] R. Williams, C. P. Gomes, and B. Selman, "Backdoors to typical case complexity," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann Publishers Inc., 2003, pp. 1173–1178.
- [44] P. Kilby, J. Slaney, S. Thiébaux, and T. Walsh, "Backbones and backdoors in satisfiability," in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2005, pp. 1368–1373.
- [45] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, vol. 7, no. 3, pp. 201–215, 1960.
- [46] C. P. Gomes, H. Kautz, A. Sabharwal, and B. Selman, "Satisfiability solvers," *Foundations of Artificial Intelligence*, vol. 3, pp. 89–134, 2008.
- [47] A. Biere, M. Heule, and H. van Maaren, *Handbook of Satisfiability: Second Edition*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. [Online]. Available: <https://books.google.com/books?id=dUAvEAAAQBAJ>
- [48] B. Selman, H. A. Kautz, and B. Cohen, "Noise strategies for improving local search," in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*. Menlo Park, CA, USA: AAAI Press, 1994, pp. 337–343.
- [49] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using QUBO models," *arXiv preprint arXiv:1811.11538*, 2018, available at <https://arxiv.org/abs/1811.11538>.
- [50] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.
- [51] N. Chancellor, S. Zohren, P. A. Warburton, S. C. Benjamin, and S. Roberts, "A direct mapping of max K-SAT and high-order parity checks to a chimera graph," *Scientific Reports*, vol. 6, no. 1, p. 37107, 2016.
- [52] M. Booth, S. Reinhardt, and A. Roy, "Partitioning optimization problems for hybrid classical/quantum execution," D-Wave Systems Inc., Burnaby, Canada, Technical Report, 2017.
- [53] G. Bass, M. Henderson, J. Heath, and J. Dulny III, "Optimizing the optimizer: Decomposition techniques for quantum annealing," *Quantum Machine Intelligence*, vol. 3, no. 1, p. 10, 2021.
- [54] F. Glover, "Tabu search—part i," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.
- [55] B. Mazure, L. Sais, and É. Grégoire, "Tabu search for sat," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI) and Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI)*. AAAI Press, 1997, pp. 281–285.
- [56] G. Palubeckis, "Multistart tabu search strategies for the unconstrained binary quadratic optimization problem," *Annals of Operations Research*, vol. 131, no. 1, pp. 259–282, 2004.
- [57] D-Wave Systems Inc., "dwave-tabu: An optimized tabu search solver for qubo problems," <https://pypi.org/project/dwave-tabu/>, 2022, version 0.5.0.
- [58] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*. Springer, 1983, pp. 466–483.
- [59] W. F. Dowling and J. H. Gallier, "Linear-time algorithms for testing the satisfiability of propositional horn formulae," *Journal of Logic Programming*, vol. 1, no. 3, pp. 267–284, 1984.
- [60] M. Järvisalo, M. J. H. Heule, and A. Biere, "Blocked clause elimination," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, vol. 6015. Springer, 2010, pp. 129–144.
- [61] B. for Anonymous Review, "Blinded for Anonymous Review," 2025, under review.
- [62] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.
- [63] H. H. Hoos and T. Stützle, "Satlib: An online resource for research on sat," in *SAT 2000: Highlights of Satisfiability Research in the Year 2000*, I. P. Gent, H. van Maaren, and T. Walsh, Eds. IOS Press, 2000, pp. 283–292.
- [64] W. Regli, G. Mossi, M. T. Hajiaghayi, H. Munoz Bauza, I. J. Whitehouse, K. Banihashem, P. Jabbarzade, and T. H. Paul, "Sat benchmarks to assess quantum-inspired solvers," GitHub repository, 2025, uMD-ARLIS / QuICC-SAT-Datasets. [Online]. Available: <https://github.com/UMD-ARLIS/QuICC-SAT-Datasets/tree/main>
- [65] J. Singer, I. P. Gent, and A. Smaill, "Backbone fragility and the local search cost peak," *Journal of Artificial Intelligence Research*, vol. 12, pp. 235–270, 2000.
- [66] M. Mosca, J. M. V. Basso, and S. R. Verschoor, "On speeding up factoring with quantum SAT solvers," *Scientific Reports*, vol. 10, no. 1, p. 15022, 2020.
- [67] J. Ding, G. Spallitta, and R. Sebastiani, "Effective prime factorization via quantum annealing by modular locally-structured embedding," *Scientific Reports*, vol. 14, no. 1, p. 3518, 2024.
- [68] K. Jun and H. Lee, "HUBO and QUBO models for prime factorization," *Scientific Reports*, vol. 13, no. 1, p. 10080, 2023.
- [69] S. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, "Quantum annealing for prime factorization," *Scientific Reports*, vol. 8, no. 1, p. 17667, 2018.
- [70] E. Andriyash, Z. Bian, F. Chudak, M. Drew-Brook, A. D. King, W. G. Macready, and A. Roy, "Boosting integer factoring performance via quantum annealing offsets," *D-Wave Technical Report Series*, vol. 14, no. 2016, p. 52, 2016.