## License

NASA/ARLIS/University of Maryland QuICC Phase I Datasets

VERSION v2025-07-31.1

This dataset is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0).

https://creativecommons.org/licenses/by/4.0/

This dataset was created under the support of the Defense Advanced Research Projects Agency (DARPA), Microsystems Technology Office (MTO), Quantum-Inspired Classical Computing Program (QuICC) and conducted during the period 2023-2025 for Phase I of this program during the period 2023-2025.

Work was performed by the members of the QuICC Test and Evaluation team consisting of employees, faculty and students at the following organizations:

* The NASA Ames Research Center  
  Quantum Artificial Intelligence Laboratory (QuAIL)
* The University of Maryland at College Park  
  Applied Research Laboratory for Intelligence and Security (ARLIS),  
  Department of Computer Science, and  
  The University of Maryland Institute for Advanced Computer Studies (UMIACS)

You are free to:

* Share — copy and redistribute the material in any medium or format
* Adapt — remix, transform, and build upon the material for any purpose, even commercially under the condition of attribution. You must give appropriate credit, provide a link to the license, and indicate if changes were made

## Citation

If you use any aspect of this this dataset, please cite it as:

SAT Benchmarks to Assess Quantum-Inspired Solvers, William Regli, Gianni Mossi, Mohammad Taghi Hajiaghayi, Humberto Munoz Bauza, Ian Joseph Whitehouse, Kiarash Banihashem, Peyman Jabbarzade, Taylor Henry Paul, 2025. Licensed under CC BY 4.0.

These files each contain a single instance of a Boolean formula. Most of the files are in conjunctive normal form (CNF). The format for these files is the community standard used by the annual SAT competitions and by DIMACS (more information is available at <https://satcompetition.github.io>).

Authoritative source for these files is   
<https://github.com/orgs/UMD-ARLIS/repositories>

## END

**Datasets for Boolean Satisfiability Problem**

This collection of SAT problems was created for the Defense Advanced Research Projects Agency (DARPA) in support of the Quantum-Inspired Classical Computing (QuICC) program Phase I. Phase I program objectives were to demonstrate a hardware solution that could solve 50 variable SAT problems natively. Based on the data and metrics obtained from these 50 variable experiments, the program performers were to extrapolate the performance of their solutions on problems of size 200 variables. The batches of SAT instances in this collection are created to benchmark these quantum-inspired hardware solvers. Hence, most batches provided focus on problems of 50 variables, with some of greater sizes up to batches with 200 variables. It has been noted in the literature on SAT that the problems of greatest computational difficulty are those that have a ratio of SAT clauses to SAT variables (clauses/variables) of approximately 4.27 (this ratio is commonly referred to as “alpha”)---hence some of the generated instances in these batches focus on ratios of vars-to-clauses that are near this value. Finally, some of these SAT instances incorporate “scale-free” (i.e. power-law) association between variables to better emulate realistic SAT problems than fully random k-SAT problem instances.

The goal of the evaluation team was to estimate the “time to solution” (TTS) and “energy to solution” (ETS) required by the various hardware approaches. Baselines for TTS and ETS were computed using State-of-the-Art SAT solvers (MiniSAT, WalkSAT, KISSAT, DPLL) running on a known hardware platform. Hence, each batch was designed to stress-test various aspects of these novel hardware systems.

Time-to-Solution (TTS) and Energy-to-Solution (ETS): TTS and ETS are respectively the time and energy to find the solution of a given instance with a 99% probability, and it is computed by using repeated runs of a single instance. Due to variations in how these SoA solvers function, as well as their initialization parameters, the TTS and ETS measurements for a given instance (and across the instances within a batch) may vary widely. To achieve some a consistency, these instances were designed to be run multiple times (>30 runs, sometimes up to 1000 trials or whatever threshold was needed to reach a degree of statistical confidence in the expected TTS/ETS for a given instance with a SoA solver) with the mean and median TTS and ETS calculations used to create a statistical profile of the difficulty for each SAT problem. Data across the instances in a batch can be used to baseline the difficulty of a given batch of SAT instances. For example, by examining the 50%-90% quantile of TTS / ETS for the whole batch one can better understand the impact of easy / hard instances on the difficulty of a given batch.

The notion of hardness of SAT instances is not completely well-defined in that a given instance may be hard for one algorithm and easier for another one; or the random seed starting a given algorithm and the nature of the search process (often stochastic or heuristic) make solutions easier to find. Hence, underlying these datasets is the assumption that the runtimes of the standard solvers are an empirical, yet objective, measurement of the difficulty of specific instances. By running various SoA solvers across instances for multiple runs, one might achieve a degree of statistical significance regarding these empirical measurements of TTS and ETS. Hence, while performing these measurements, various properties of instances and batches were revealed, i.e., certain instances had a very tight consistency of TTS and ETS, others had a much greater spread; some instances favored the approximate solvers (WalkSAT) while others favored the complete solvers (MiniSAT); etc. Metrics created using the SoA solvers were then used to compare/contrast with the metrics gathered by the DARPA QuICC performers.

The datasets include batches of consistently generated instances resulting from increasing the number of variables, enabling scaling of the TTS / ETS and projections for hardware performance and scale up. The 50%-90% quantile of TTS / ETS can be used to determine the performance of the hardware or hardware simulators simulator for these larger sizes.

In tracking TTS / ETS across batches the team estimated the number of instances solved for a fixed time or fixed energy allocation. This measure provided an empirical estimate of an approximation factor or accuracy-vs-time/accuracy-vs-energy trade off. This also allows estimating performance given possible constraints on the amount of available resources.

Descriptions of instances in these files and directories is provided below.

Additional information on the batches, as well as metrics created with state-of-the-art solvers (2025), is provided as part of the technical report: SAT Benchmarks to Assess Quantum-Inspired Solvers, William Regli, Gianni Mossi, Mohammad Taghi Hajiaghayi, Humberto Munoz Bauza, Ian Joseph Whitehouse, Kiarash Banihashem, Peyman Jabbarzade, Taylor Henry Paul, 2025.

All instances of the SAT problem in the batches of this collection are formatted using the same syntax as recommended for the annual SAT Competition (https://satcompetition.github.io/).

A brief description of each collection of batches is provided.

**Batches batch-01 to batch-04: AI Planning**

Henry Kautz and Bart Selman showed in 1990 that AI planning had equivalences to the Boolean Satisfiability problem (<https://www.cs.cornell.edu/selman/papers/pdf/92.ecai.satplan.pdf>). While it is possible to mapping STRIPS-style planning instances to a SAT problem, the size of the SAT problem tends to get very large. For example, for simple blocks-world problems of 4-to-5 blocks, several hundred variables and over a thousand clauses may be required. Blocks world problems of 3 blocks typically would have about 40-60 variables and about 250 clauses. The instances in these batches are random SAT instances, all of which are satisfiable, that are generated to align with the structure of blocks-world planning problems that have been encoded as SAT. The alpha ratio for many of these instances is between 4-4.5.

Batch Name # instances Notes

batch-01 101 Approximate complexity to 3 block blocks-world

batch-02 251 Approximate complexity to 3 block blocks-world

batch-03 51 Approximate complexity to 3 block blocks-world

batch-04 101 Approximate complexity to 3 block blocks-world

**Batches batch-05 to batch-06, batch-09 to batch-20, bach-25 to batch-28:   
Planted Solutions**

Planted solutions in Boolean satisfiability (SAT) refer to instances intentionally constructed by embedding one or more known solutions within an otherwise random or semi-random SAT formula. Such instances are typically generated by first selecting a particular assignment of Boolean variables (the "planted solution") and then creating clauses consistent with that solution. Since the instances are generated algorithmically, the number of instances can easily be adjusted. The resulting SAT instance thus has at least one guaranteed satisfying assignment. Regarding the difficulty of instances, some of the parameters used for generating the instance do indeed serve as proxies for hardness. For instance, for the planted instance with multiple solutions, a higher number of solutions that are planted generally implies that the decision problem is easier. Hence, planted solutions provide controlled, realistic benchmarks essential for analyzing and understanding SAT solver performance and complexity. Despite guaranteeing at least one satisfying assignment, their construction typically creates subtle, challenging instances, bridging theory, experimental study, and practical solver improvements.

In the context of QuICC benchmarking, these instances provide a structured approach for analyzing average-case computational complexity of SAT algorithms. Random SAT instances often form the basis of complexity-theoretic benchmarks, and planted instances offer intermediate control between purely random instances and structured real-world instances.

The instances in batches below all involve planted solutions; with batches differentiating instances based on the number of planted solutions and the number of clauses being used.

Batch Name # instances Notes

batch-05 50 Quiet-planting 4-SAT (UNSAT phase)

batch-06 50 Quiet-planting 4-SAT (UNSAT phase)

batch-09 101 Quiet-planting 6-SAT (UNSAT phase)

batch-10 101 Quiet-planting 7-SAT (UNSAT phase)

batch-11 50 Statistically hard instances of 4SAT with

3 - 6 planted solutions; 10 variables, ~100 clauses

batch-12 50 Statistically hard instances of 4SAT with

3 - 6 planted solutions; 20 variables, ~200 clauses

batch-13 50 Statistically hard instances of 4SAT with

3 - 6 planted solutions; 40 variables, ~500 clauses

batch-14 50 Statistically hard instances of 6SAT with

7 - 10 planted solution; 10 variables, ~500 clauses

batch-15 50 Statistically hard instances of 6SAT with

7 - 10 planted solution; 20 variables, ~1000 clauses

batch-16 50 Statistically hard instances of 7SAT with

7 - 10 planted solution; 10 variables, ~1200 clauses

batch-17 50 Statistically hard instances of 4SAT with

at least 2 planted solution; 125 variables, ~1300 clauses

batch-18 50 Statistically hard instances of 4SAT with

at least 2 planted solution; 150 variables, ~1570 clauses

batch-19 50 Statistically hard instances of 4SAT with

at least 2 planted solution; 175 variables, ~1800 clauses

batch-20 50 Statistically hard instances of 4SAT with

at least 2 planted solution; 200 variables, ~2100 clauses

batch-25 115 Statistically hard instances of 4SAT with

1 planted solution, 80 clauses (10 variables)

batch-26 101 Statistically hard instances of 4SAT with

1 planted solution, 120 clauses (20 variables).

batch-27 101 Statistically hard instances of 4SAT with

1 planted solution, 30 clauses (180 variables)

batch-28 101 Statistically hard instances of 4SAT with

1 planted solution, 240 clauses (40 variables)

**Batches batch-07 to batch-08, batch-21 to batch-24: SemiPrime factorization**

SemiPrime factorization in SAT solving refers to the encoding of factoring problems into Boolean satisfiability (SAT) problems, specifically targeting numbers known as semiprimes—numbers that are the product of exactly two prime numbers. This transformation encodes the integer factorization problem as a SAT instance, where solutions to the SAT instance correspond directly to factors of the given semiprime. SemiPrime factorization underpins the security of cryptographic schemes (notably RSA encryption), hence factorization benchmarks reflect practical complexity relevant to cryptanalysis, providing benchmarks that have tangible real-world significance.

SemiPrime factorization has several characteristics making it attractive as a SAT-solving benchmark, including the following.

1. Structural Complexity: Factorization encodes arithmetic operations into logical clauses, resulting in structured yet highly nontrivial instances. This complexity challenges solvers that exploit structural patterns and test their ability to handle arithmetic-like structures efficiently.
2. Scalable Difficulty: Adjusting the bit length of the semiprime controls the instance difficulty smoothly and predictably. Small numbers factor easily, but the difficulty grows sharply and predictably as number size increases, providing clear metrics for solver performance.
3. Verification Simplicity: Semiprime numbers used as benchmarks often have known factorizations. Thus, these instances have an embedded "planted solution," allowing straightforward verification of solver correctness and efficiency.
4. Robustness against Heuristics: Factorization SAT instances typically exhibit low-level, arithmetic-specific structures that are difficult to exploit with generic heuristics. This property tests solver generality, adaptability, and ability to handle computationally challenging instances.

The instances in batches below are based on semi-prime factorization problems; with batches differentiating instances based on the number of planted solutions and the number of clauses being used.

Batch Name # instances Notes

batch-07 105 Reduction from semiprime integer factorization

batch-08 258 Reduction from semiprime integer factorization

batch-21 50 Semi-primes instances with n={19,20}; ~300 variables,

~1500 clauses, and on average 3.31 literals per clause

batch-22 50 Semi-primes instances with n={29,30}; ~650 variables,

~3500 clauses, and on average 3.31 literals per clause

batch-23 50 Semi-primes instances with n={39,40}; ~1200 variables,

~6500 clauses, and on average 3.31 literals per clause

batch-24 50 Semi-primes instances with n={49,50}; ~1850 variables,

~10000 clauses, and on average 3.31 literals per clause

**Batches batch-29 to batch-38: Maximum Likelihood Decoding**

The McEliece/Niederreiter cryptosystems and the related problem of Maximum Likelihood Decoding (MLD) or Maximum Likelihood Estimation (MLE) in coding theory have been increasingly studied in the context of SAT solving and benchmarking due to their rich combinational structure, cryptographic importance, and theoretical significance. These problems are cryptographic schemes considered as post-quantum secure candidates. In benchmarking SAT solvers with these problems one can provide insights directly relevant to cryptographic research, security analysis, and cryptanalysis. The McEliece/Niederreiter cryptosystems and related MLD problems offer a highly structured, scalable, and cryptographically significant domain for SAT solving and benchmarking. They enable comprehensive testing of solver capabilities in handling algebraic and arithmetic constraints, supporting advanced SAT-solving research, cryptanalysis efforts, and broader implications for post-quantum cryptography research. These problem instances can also be formulated mixing both disjunctive and XOR clauses.

In the batches below, instances of the McEliece/Niederreiter cryptosystem defined as a Maximum Likelihood Detection where H is the public key and y is the encrypted message. Public keys are generated using random Goppa codes, and the plaintext messages x are randomly generated. For more information please see Generating Hard Ising Instances With Planted Solutions Using Post-Quantum Cryptographic Protocols (<https://arxiv.org/abs/2308.09704>).

Batch Name # instances Notes

batch-29 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t3 (n=40) 632-720 variables.

batch-30 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t3 (n=44) 786-878 variables.

batch-31 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t3 (n=48) 920-1038 variables

batch-32 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t4 (n=48) 820-930 variables

batch-33 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t4 (n=52) 1004-1106 variables

batch-34 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t4 (n=56) 1174-1278 variables

batch-35 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t5 (n=56) 1022-1106 variables

batch-36 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t5 (n=60) 1216-1342 variables

batch-37\* 100 All \*\_3sat.cnf reductions in McEliece

Batch 2 / t5 (n=64) 1420-1548 variables.

batch-38\* 100 All \*\_3sat.cnf reductions in McEliece

Batch 1 / t3 (n=16) 218-272 variables.

**Batch N\_eq\_50\_seen (100 instances total)**

This batch consists of instances for the QuICC performers to solve using their HARDWARE. It contains 100 instances from our previously shared example batches (01-38). For selecting these instances, we used a script to compute the minimum, maximum, and average of the MiniSAT running time of each instance across all batches with 50 variables (Batches 01-05, 07, 09, 10-16, 25-28). Sorting the instances on average runtime, we selected approximately uniformly across the distribution of difficulties.

**Batch N\_eq\_50 (950 instances total):**

This collection of batches consists of instances used at the end of Phase I by QuICC performers to test using their HARDWARE. It contains instances different from N\_eq\_50\_seen as well as batches 01-38.

Batch Name # instances Notes

batch\_0 400 4 sub-batches of 100 instances each of random 2-SAT with

N = 50, M = 60 (all satisfiable). Batches are labelled from

easiest (C0) to hardest (C3) according to WalkSAT performance

batch\_1 400 4 sub-batches of 100 instances each of scale-free random 3-SAT

with N = 50, M = 200, \mu = 0.4 (all satisfiable). Batches are

labelled from easiest (C0) to hardest (C3) according to

WalkSAT performance

batch\_2 50 Random K=3 SAT, N=50 M = 225: Lowest 25% by Walksat

iterations to solution, calculated using 1000 repetitions pe

instance. All satisfiable. Backbones and number of solutions

for each instance is known.

batch\_3 50 Random K=3 SAT, N=50 M = 225: Highest 25% by Walksat

iterations to solution, calculated using 1000 repetitions per instance. All satisfiable. Backbones and number of solutions known.

batch\_4 50 Random K=4 SAT, N=40 M = 400: Lowest 66% by Walksat

iterations to solution, calculated using 1000 repetitions per

instance. All satisfiable. Backbones and number of solutions

known.

**Batch N\_gt\_50 (932 instances total)**

The goal of batch N\_gt\_50 was to provide instances with greater numbers of variables to test the performers’ projections, via simulation and emulation, of their hardware performance as the size of the problem scales up.

Batch Name # instances Notes

batch\_0 400 4 sub-batches of 100 instances each of random 2-SAT with

N = 200, M = 240 (all satisfiable). Batches are labelled from

easiest (C0) to hardest (C3) according to WalkSAT performance

batch\_1 400 4 sub-batches of 100 instances each of scale-free random 3-SAT

with N = 200, M = 800, \mu = 0.4 (all satisfiable). Batches are

labelled from easiest (C0) to hardest (C3) according to WalkSAT

performance

batch\_2 82 Semiprime-factorization instances of various sizes up to N=200

batch\_3 50 Random K=3 SAT, N=200, M=840: Lowest 25% by Walksat

iterations to solution, calculated using 200 repetitions per

instance. All satisfiable (post-selected with Walksat). No more than around 300,000 walksat iterations to solve with 99% probability.

**Test Batches**

These batches of instances were created as part of preparation for the release of the final set for QuICC Phase I benchmarking and consist of subsets of instances from the N\_eq\_50, N\_eq\_50\_seen and N\_gt\_50 batches.

**50-var instances**

Batch Name # instances Notes

t\_batch50\_0 100 C3 subset of batch 0 of N\_eq\_50

(100 “harder” 2-SAT instances)

t\_batch50\_1 100 C3 subset batch 1 of N\_eq\_50

(100 “harder” scale-free 3-SAT instances)

t\_batch50\_2 100 Batch 2 + batch 3 of N\_eq\_50

(100 uniformly-random 3-SAT instances)

t\_batch50\_3 50 Batch 4 of N\_eq\_50

(50 uniformly-random 4-SAT instances)

t\_batch50\_4 100 the full N\_eq\_50\_seen directory

(100 selected instances from old batches)

**Up to 200-var instances**

Batch Name # instances Notes

t\_batch200\_0 20 formulas 0-19 of C0 subset of batch 0 of N\_gt\_50

(20 “easier” 2-SAT instances)

t\_batch200\_1 20 formulas 0-19 of C0 subset of batch 1 of N\_gt\_50

(20 “easier” scale-free 3-SAT instances)

t\_batch200\_2 82 all of batch 2, different sizes of N\_gt\_50

(82 semiprime instances)

t\_batch200\_3 20 formulas 000-019 of batch 3 of N\_gt\_50

(20 3-SAT instances)