Minimum Disagreement Parity (MDP) Benchmark

Randal E. Bryant
Computer Science Department
Carnegie Mellon University, Pittsburgh, PA, United States
Email: Randy.Bryant@cs.cmu.edu

I. Obtaining Benchmarks

The benchmark generator, files, and documentation are available at:

https://github.com/rebryant/mdp-benchmark

II. Description

Crawford, Kearns, and Shapire proposed the Minimum Disagreement Parity (MDP) Problem as a challenging SAT benchmark in an unpublished report from AT&T Bell Laboratories in 1994 [1].

MDP is closely related to the "Learning Parity with Noise" (LPN) problem. LPN has been proposed as the basis for public key crytographic systems [2]. Unlike the widely used RSA cryptosystem, it is resistant to all known quantum algorithms [3]. The capabilities of SAT solvers on MDP is therefore of interest to the cryptography community. We provide these benchmarks as a way to stimulate the SAT community to expand beyond pure CDCL, incorporating other solution methods into their SAT solvers.

Crawford wrote a benchmark generator in the 1990s and supplied several files to early SAT competitions with names of the form par$N$-S.cnf, where N is the size parameter and S is a random seed. These had values of $N \in \{8, 16, 32\}$. These files are still available online at https://www.cs.ubc.ca/ hoos/SATLIB/benchm.html. SAT solvers of that era were challenged by $N = 16$ and could not possibly handle $N = 32$. Unfortunately, the code for his benchmark generator has disappeared.

We wrote a new benchmark generator for the MDP problem. In doing so, we added more options for problem parameters and encoding methods. We also replaced the binary encoding of at-most-k constraints with a more SAT-solver-friendly unary counter encoding [4].

III. Problem Description

In the following, let $\mathcal{B} = \{0, 1\}$ and $\mathcal{N}_p = \{1, 2, \ldots, p\}$. Assume all arithmetic is performed modulo 2. Thus, if $a, b \in \mathcal{B}$, then $a + b \equiv a \oplus b$.

The problem is parameterized by a number of solution bits n, a number of samples m, and an error tolerance k, as follows. Let $\boldsymbol{s} = s_1, s_2, \ldots, s_n$ be a set of *solution* bits. For $1 \leq j \leq m$, let $X_j \subseteq \mathcal{N}_n$ be a *sample set*, created by generating n random bits $x_{1,j}, x_{2,j}, \ldots x_{n,j}$ and letting $X_j = \{i | x_{i,j} = 1\}$. Let $\boldsymbol{y} = y_1, y_2, \ldots, y_m$ be the parities of the solution bits for each of the m samples:

$$y_j \quad = \quad \sum_{i \in X_j} s_i \qquad (1)$$

Given enough many samples m for there to be at least n linearly independent sample sets, the values of the solution bits $\boldsymbol{s}$ can be uniquely determined from $\boldsymbol{y}$ and the sample sets S_j for $1 \leq j \leq m$ by Gaussian elimination. To make this problem challenging, we introduce "noise," allowing up to k of these samples to be "corrupted" by flipping the values of their parity. That is, let $T \subseteq \mathcal{N}_m$ be created by randomly choosing k values from $\mathcal{N}_m$ without replacement, and define m "corruption" bits $\boldsymbol{r} = r_1, r_2, \ldots, r_m$, with r_j equal to 1 if $j \in T$ and equal to 0 otherwise. We then provide noisy samples $\boldsymbol{y}'$, defined as:

$$y'_j \quad = \quad r_j + \sum_{i \in X_j} s_i \qquad (2)$$

and require the correct solution bits $\boldsymbol{s}$ to be determined despite this noise. That is, the generated solution $\boldsymbol{s}$ must satisfy at least $m - k$ of equations (1). For larger values of k, the problem becomes NP-hard.

This problem can readily be encoded in CNF with variables for unknown values $\boldsymbol{s}$ and $\boldsymbol{r}$, along with some auxilliary variables. We further parameterize the problem with a value $t \leq k$, indicating the maximum number of corrupted samples accepted in the solution, where the problem should be satisfiable when $t = k$ but may become unsatisfiable for $t = k - 1$. Each of the m equations (2) is encoded using auxilliary variables to avoid exponential expansion. An at-most-t constraint is imposed on the corruption bits $\boldsymbol{r}$.

For $t = k$, the solution $\boldsymbol{s}$ is not guaranteed to be unique, but we allow any solution that satisfies at least $m - k$ of the constraints (1). In addition, setting $t = k - 1$ does not guarantee that the formula is unsatisfiable. Indeed, we found some instances where there was a solution that satisfied $m - k + 1$ constraints.

By analyzing the CNF file, it is possible to discern the sample sets S_j and the values of the noisy samples $\boldsymbol{y}'$. The values of $\boldsymbol{s}$ and $\boldsymbol{r}$, however, remain hidden, except as comments at the start of the file.

Crawford suggests choosing n to be a multiple of 4 and letting $m = 2n$ and $k = m/8 = n/4$. Our benchmark files were all generated under those conditions.

IV. PROVIDED FILES

The generator `mdp-gen.py` and an associated README file are located in the `src` subdirectory. This directory also contains a program `mdp-check.py`. Given a `.cnf` file generated by `mdp-gen.py` and the output of a successful run of a SAT solver, it can check that the solution for the input variables representing $\boldsymbol{s}$ indeed satisfies at least $m - k$ of the equations of (1). The supplied benchmark files were generated by running the script `generate.sh` in the `src` subdirectory.

There are 30 files: five satisfiable and five unsatisfiable instances for $n \in \{28, 32, 36\}$, generated using five different random seeds. The seeds were adjusted so that the 15 instances generated with $t = k - 1$ are all unsatisfiable, as is discussed below.

We tested these benchmarks using the KISSAT [5] CDCL solver. Measurements were peformed on a 3.2 GHz Apple M1 Max processor with 64 GB of memory and running the OS X operating system, with a time limit of 5000 seconds per run. For $n = 28$, KISSAT can easily handle both the satisfiable and the unsatisfiable instances, with times ranging from 30 to 900 seconds. For the satisfiable instances with $n = 32$, it can solve some in just 30 seconds, but times out for two of the five runs. It times out on all unsatisfiable instances for $n = 32$, and it times out on all satisfiable and unsatisfiable instances for $n = 36$.

We also tested the benchmarks with CRYPTOMINISAT, a CDCL solver augmented with the ability to perform Gauss-Jordan elimination on parity constraints [6]. It can easily handle all satisfiable instances, never requiring more than 90 seconds. When not required to generate a proof of unsatisfiability, it can also easily handle all of the unsatisfiable instances. That is how we ensured that the instances with $t = k - 1$ are unsatisfiable. Currently, CRYPTOMINISAT cannot generate DRAT proofs of unsatisfiability when it uses Gauss-Jordan elimination, and so it fares no better than KISSAT on the unsatisfiable instances when proof generation is required. CRYPTOMINISAT can scale to $n = 60$ without difficulty. Nonetheless, the problem is still NP-hard, and so even CRYPTOMINISAT only pushes the boundary before exponential scaling limits feasibility.

REFERENCES

[1] J. M. Crawford, M. J. Kearns, and R. E. Schapire, "The minimal disagreement parity problem as a hard satisfiability problem," 1994. [Online]. Available: http://www.cs.cornell.edu/selman/docs/crawford-parity.pdf

[2] J. Katz, "Efficient cryptographic protocols based on the hardness of learning parity with noise," in *Cryptography and Coding*, ser. LNCS, vol. 4887, 2007, pp. 1–15.

[3] K. Pietrzak, "Cryptography from learning parity with noise," in *SOFSEM 2012: Theory and Practice of Computer Science*, ser. LNCS, vol. 7147, 2012, pp. 99–114.

[4] C. Sinz, "Towards an optimal CNF encoding of Boolean cardinality constraints," in *Principles and Practice of Constraint Programming (CP)*, ser. LNCS, vol. 3709, 2005, pp. 827–831.

[5] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[6] M. Soos, K. Nohl, and C. Castelluccia, "Extending SAT solvers to cryptographic problems," in *Proc. of the 12th Int. Conference on Theory and Applications of Satisfiability Testing* (*SAT 2009*), ser. LNCS, vol. 5584, 2009, pp. 244–257.