Minimum Disagreement Parity (MDP) Benchmark*

Randal E. Bryant

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

1 Obtaining Benchmarks

The benchmark generator, files, and docmumentation is available at:

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

2 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Kearns, and Shapire proposed the Minimum Disagreement Parity (MDP) Problem as a challenging SAT benchmark in an unpublished report from AT&T Bell Laboratories in 1994 [2].

MDP is closely related to the "Learning Parity with Noise" (LPN) problem. LPN has been proposed as the basis for public key crytographic systems [3]. Unlike the widely used RSA cryptosystem, it is resistant to all known quantum algorithms [4]

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 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 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 [5].

3 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

* This work was supported by the U. S. National Science Foundation under grant CCF-2108521

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:

$$y_j = \sum_{i \in X_j} s_i \tag{1}$$

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:

$$y'_j = r_j + \sum_{i \in X_j} s_i \tag{2}$$

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). Setting $t = k - 1$ does not guarantee that the formula is unsatisfiable, although we found this to be the case for all of the instances we tested.

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.

4 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.

We tested these benchmarks using the CDCL solver KISSAT [1]. Measurements were peformed on a 3.2 GHz Apple M1 Max processor with 64 GB of memory and running the OS X operating system. KISSAT can easily handle all instances with $n = 28$. For $n = 32$, its times range for XX to YY seconds for the satisifiable instances and XX to YY seconds for the unsatisfiable instances. It could not complete any of the instances for $n = 36$ within a 5000-second time limit.

We also tested the satisfiable instances with CRYPTOMINISAT, a CDCL solver augmented with the ability to perform Gauss-Jordan elimination on parity constraints [6]. It can easily handle all instances. Indeed, it can scale to $n = 60$ without difficulty. Nontheless, the problem is still NP-hard, and so even CRYPTOMINISAT only pushes the boundary before exponential scaling limits feasibility. 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.

References

1. Biere, A., Fazekas, K., Fleury, M., Heisinger, M.: CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In: Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions. Department of Computer Science Report Series B, vol. B-2020-1, pp. 51–53. University of Helsinki (2020)
2. Crawford, J.M., Kearns, M.J., Schapire, R.E.: The minimal disagreement parity problem as a hard satisfiability problem (1994), `http://www.cs.cornell.edu/selman/docs/crawford-parity.pdf`
3. Katz, J.: Efficient cryptographic protocols based on the hardness of learning parity with noise. In: Cryptography and Coding. LNCS, vol. 4887, pp. 1–15 (2007)
4. Pietrzak, K.: Cryptography from learning parity with noise. In: SOFSEM 2012: Theory and Practice of Computer Science. LNCS, vol. 7147, pp. 99–114 (2012)
5. Sinz, C.: Towards an optimal CNF encoding of Boolean cardinality constraints. In: Principles and Practice of Constraint Programming (CP). LNCS, vol. 3709, pp. 827–831 (2005)
6. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Proc. of the 12th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2009). LNCS, vol. 5584, pp. 244–257 (2009)