Generating formulas/programs with RandomGenerator

Giovanni Amendola[1], Francesco Ricca[1], Miroslaw Truszczynski[2]

[1]Department of Mathematics and Computer Science, University of Calabria, Italy
{amendola,ricca}@mat.unical.it
[2]Department of Computer Science, University of Kentucky, USA
mirek@cs.uky.edu

Abstract. This document presents a strategy and some useful results for using our random generator of formulas QSAT in non-clausal form, and disjunctive logic programs.
Form more details about the random models see [3].

Keywords: Answer Set Programming, Random Boolean Formulas, Phase Transition, Random Logic Programs

1 Introduction

The ability to generate large numbers of CNF formulas and QBFs of a desired hardness is important [8]. Inherently hard instances for SAT and QBF solvers are essential for designing and testing new search techniques [1], and are used in solver competitions [9, 11, 5]. Large collections of *easy* instances support the so-called *fuzz* testing and can help reveal issues in solver implementation, as well as defects in solver design [4]. The ability to generate large numbers of easy and hard logic programs is equally important to the field of answer-set programming solvers [7, 2]. Random formulas have been used successfully to assess and test CDCL solvers on resource management, efficacy of heuristics, see e.g., [6, 10].

The generator can be used for assessing solver performance or testing an implementation, and these activities usually have different requirements for instance properties. In presence of so many generation parameters, it is not immediate to choose the right settings for the purpose. Here we provide simple guidelines for identifying the settings for generating formulas of the desired hardness. The key underlying property is that all our models show some form of the easy-hard-easy pattern that can be exploited to find "areas" of formulas of varying difficulty.

2 Parameter space exploration strategy

For the Chen-Interian model, one strategy is to fix a and e (to define the structure of a clause). Next for each pair of values of $A = |X|$ and $E = |Y|$, one runs the tool with different numbers m of clauses/rules. The formulas (programs) generated in this way show the phase transition and the corresponding easy-hard-easy pattern. Running a solver on those formulas allows one to observe these properties and select the value of m that yields formulas/programs of the desired difficulty.

3 Empirical results

Experiments were run on a Debian Linux with 2.30GHz Intel Xeon E5-4610 v2 CPUs and 128GB of RAM. Each single execution was constrained to one single core by using the *taskset* command. Time measurements were performed by using the *runlim* tool. In all the experiments the results are averaged over 64 samples of the same size. We report for space reasons and w.l.o.g. the time measurements obtained by running the well-known ASP solver clasp. Indeed, our generator allows to print the same instance of a model in several formats, and the results have been demonstrated to be reasonably solver independent [3].

The results are summarized in Figure 1-4 formulas with clauses of four variables (one universal and three existentials) both in terms of frequency of satisfiability (plots on the left) and in terms of execution time (plots on the right).

Concerning the Chen Interian Model we observe in Figure 1 the behavior of formulas with one component. The plots allow to identify the phase transition zone, and in correspondence one can find the hardest instances. As soon as the number of existential variables (E) grows (observe plots from the top to the bottom of the page) an area of maximum hardness (having the shape of a galaxy) arises in correspondence of some critical values for the ratio of existential over universal variables.

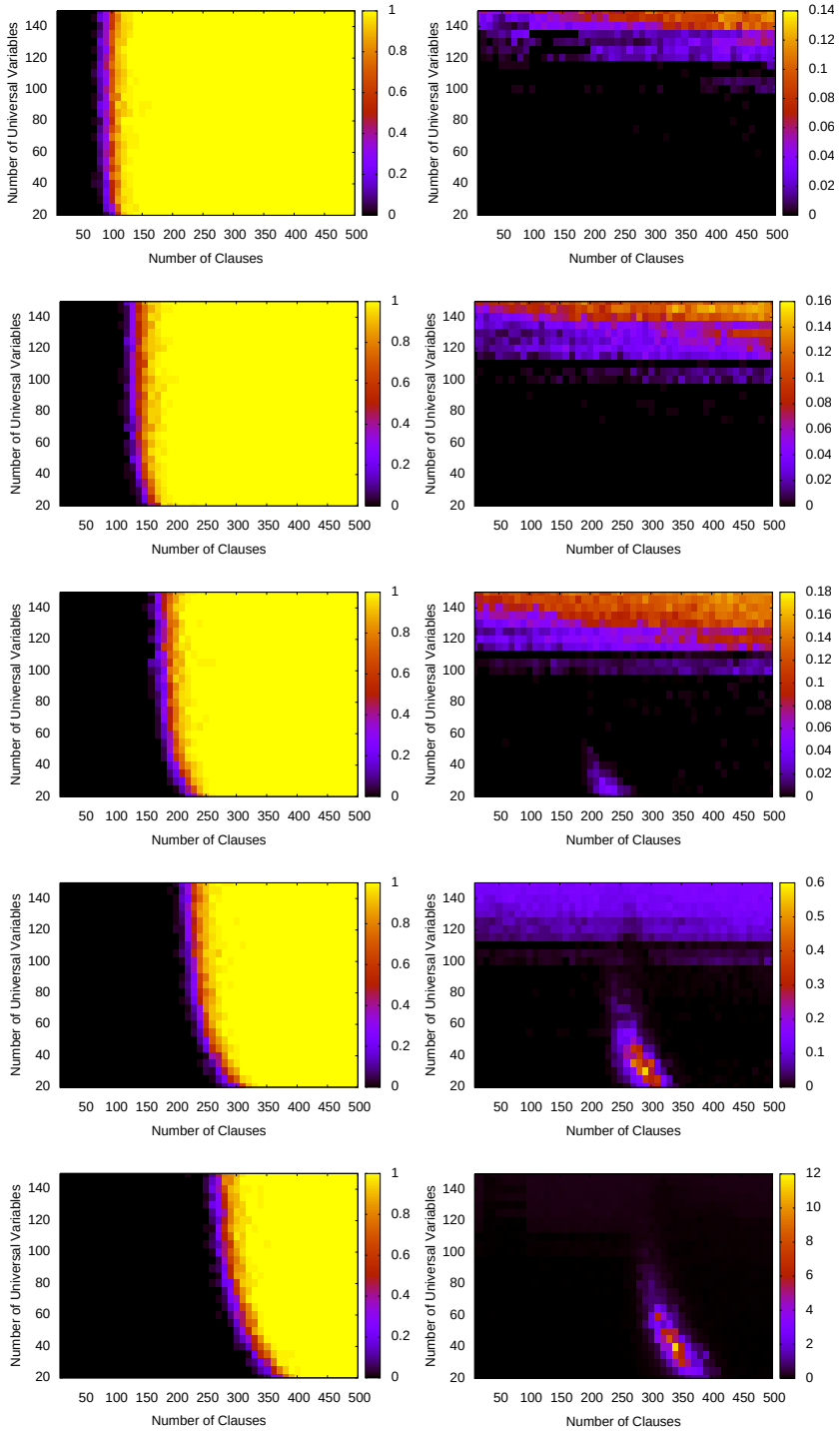
The results obtained by increasing the number of components to two are reported in Figure 2. As expected, the phase transitions move slightly on the right (as discussed in [**?**]) and hardness significantly increases. Indeed the area of maximum hardness is already visible with $E = 30$, and the peak of hardness is up to about two order of magnitude higher w.r.t. instances with $t = 1$ for $E = 60$.

The results of an analogous experiment for the controlled model are reported in Figure 4, which shows the behavior of formulas with one component. Here the number of clauses is precisely two times the number of universal variables, thus we have only one plot. The behavior of formulas from this model is more predictable, indeed hardness just grows with the number of variables along the phase transition ridge. As reported in [**?**], the hardest instances of controlled model are harder than those obtained with Chen Interian with the same number of existential variables, and this is reflected in out experiment, e.g., when $E = 60$ about 40 seconds on average are needed for solving controlled model instances whereas 10 seconds are enough for Chen Interian ones. Once one increases the number of components controlled model instances become extremely hard, making unfeasible (with current hardware/solving technology) the exploration of the same space of parameters. Nonetheless, in Figure 5 we summarize the behavior of that model just for formulas with $E = 12$ and varying number of components up to 11. Note that we already get a hardness peak at 60 seconds by generating formulas with only 11 components.

It is worth observing that the plots in Figures 1-5 can be used as a reference for selecting generation parameters. For example, if one wants very easy satisfiable instances from Chen Interian, we know from Figures 1 that 1-$Q(1,3,40,20,350)$ are such. Alternatively, medium-hard instances (that are suited for assessing industrial solvers) belong to the family 2-$Q(1,3,60,40,340)$ (see Figure 2). In case super hard formulas are needed, one can select controlled model and increase the number of components at will.

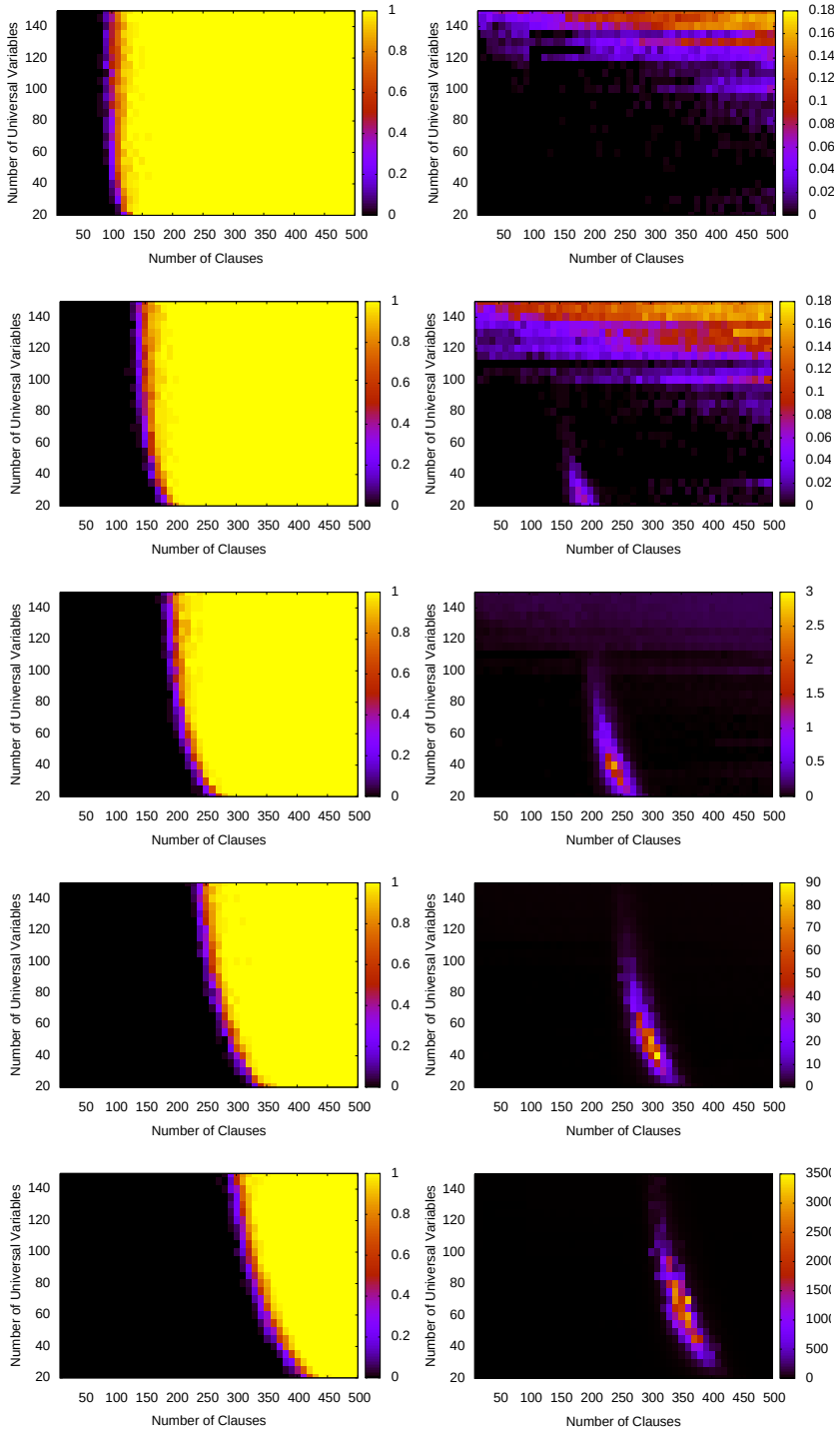
References

1. Achlioptas, D.: Random satisfiability. In: Handbook of Satisfiability, pp. 245–270 (2009)
2. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: LPNMR 2015. pp. 40–54 (2015)
3. Amendola, G., Ricca, F., Truszczynski, M.: Generating hard random boolean formulas and disjunctive logic programs. In: Proceedings of IJCAI 2017 (2017)
4. Brummayer, R., Lonsing, F., Biere, A.: Automated testing and debugging of SAT and QBF solvers. In: SAT 2010. pp. 44–57 (2010)
5. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the fifth answer set programming competition. Artif. Intell. 231, 151–181 (2016), http://dx.doi.org/10.1016/j.artint.2015.09.008
6. Elffers, J., Nordström, J., Simon, L., Sakallah, K.: Seeking practical cdcl insights from theoretical sat benchmarks. In: Presentation at the Pragmatics of SAT 2016 workshop (2016), http://www.csc.kth.se/jakobn/research/TalkPoS16
7. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: *clasp* : A conflict-driven answer set solver. In: LPNMR 2007. pp. 260–265 (2007)
8. Gent, I.P., Walsh, T.: Beyond NP: the QSAT phase transition. In: AAAI 1999. pp. 648–653 (1999)
9. Järvisalo, M., Berre, D.L., Roussel, O., Simon, L.: The international SAT solver competitions. AI Magazine 33(1) (2012), http://www.aaai.org/ojs/index.php/aimagazine/article/view/2395
10. Järvisalo, M., Matsliah, A., Nordström, J., Zivny, S.: Relating proof complexity measures and practical hardness of SAT. In: CP. Lecture Notes in Computer Science, vol. 7514, pp. 316–331. Springer (2012)
11. Narizzano, M., Pulina, L., Tacchella, A.: Report of the third QBF solvers evaluation. JSAT 2(1-4), 145–164 (2006), http://jsat.ewi.tudelft.nl/content/volume2/JSAT2_6_Narizzano.pdf



(a) Frequency of Satisfiability (b) Execution Time (s)

Fig. 1: Phase transition and Hardness in Chen-Interian model formulas 1-$Q(1,3,A,E,m)$.



(a) Frequency of Satisfiability (b) Execution Time (s)

Fig. 2: Phase transition and Hardness in Chen-Interian model formulas 2-$Q(1,3,A,E,m)$.

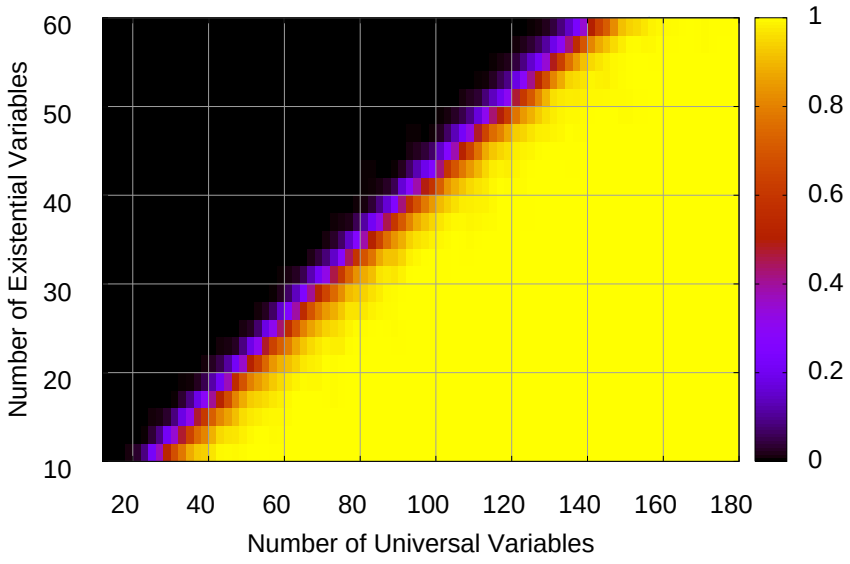
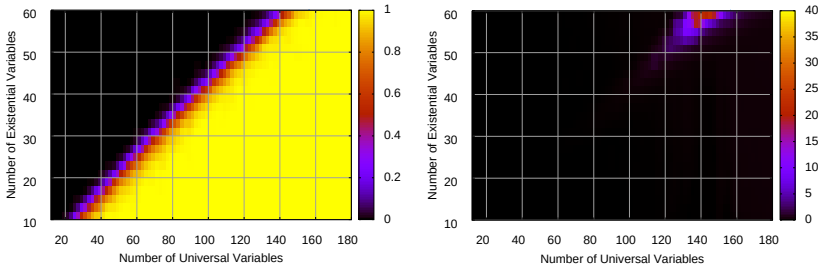


Fig. 3: The position of the phase transition in controlled model formulas $Q^{ctd}(4,A,E)$.



(a) Frequency of Satisfiability (b) Execution Time (s)

Fig. 4: Phase transition and Hardness in controlled model formulas $Q^{ctd}(4,A,E)$.

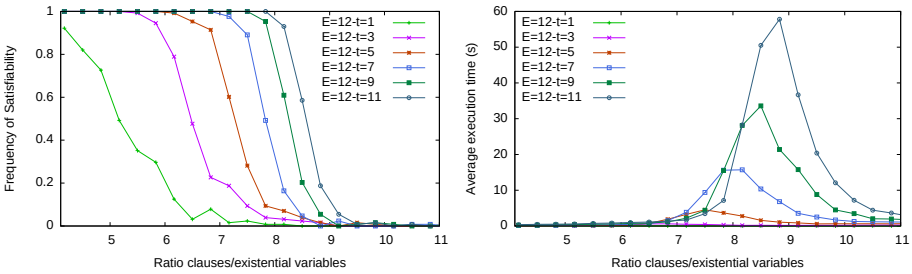


Fig. 5: Effect of varying the number of components.