Features for SAT

Lin Xu, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown

University of British Columbia,
201-2366 Main Mall, Vancouver, BC, CANADA

1 Overview

For the propositional satisfiability (SAT) problem we used the 138 features listed in Figure 1. Since a preprocessing step can significantly reduce the size of the CNF formula (especially for industrial-like instances), we chose to apply the preprocessing procedure `SATElite` [1] on all instances first, and then to compute instance features on the preprocessed instances. The first 90 features, except Features 22–31, were introduced by [5]. They can be categorized as *problem size* features (1–7), *graph-based* features (8–21), *balance* features (37–49), *proximity to horn formula* features (50–55), *DPLL probing* features (56–62), and *local search probing* features (69–90).

2 Features for Propositional Satisfiability (SAT)

We incrementally introduced additional features in our work on `SATzilla` [6, 7] over the last five years, but describe them here for the first time. Features 22–26 are related to the graph diameter [2]. For each node in the variable graph, we computed the longest shortest path between it and any other node and report statistics across all nodes. Features 32–36 are related to the clustering coefficient, a measure of local graph cliqueyness. For each node in the clause graph, let p denote the number of edges present between the node and its neighbors, and let m denote the maximum possible number of such edges; we report statistics of p/m across all nodes. Our new *clause learning* features (91–108) are based on statistics gathered in 2-second runs of `Zchaff_rand` [4]. We measured the number of learned clauses (Features 91–99) and the length of the learned clauses (Features 100–108) after every 1000 search steps, computing statistics over the resulting vectors.

Our *survey propagation* features (109–126) are derived from estimates of variable bias in a SAT formula based on probabilistic inference [3]. We used `VARSAT`'s implementation to estimate the probability that each variable is required to be true or false, or is unconstrained. Features 109–117 measure the confidence of survey propagation (that is, $\max(P(\text{true})/P(\text{false}), P(\text{false})/P(\text{true}))$ for all variables) and Features 118–126 compute the statistics of $P(\text{unconstrained})$.

Finally, our timing features (127–138) measure the time taken by 12 different blocks of feature computation code: instance preprocessing by `SATElite`; problem size, variable-clause graph and balance features (1–17, 37–49); variable graph

Problem Size Features:

1–2. **Number of variables and clauses in original formula**: denoted v and c, respectively
3–4. **Number of variables and clauses after simplification with** `SATElite`: denoted v' and c', respectively
5–6. **Reduction of variables and clauses by simplification**: (v-v')/v' and (c-c')/c'
7. **Ratio of variables to clauses**: v'/c'

Variable-Clause Graph Features:

8–12. **Variable node degree statistics:** mean, variation coefficient, min, max, and entropy
13–17. **Clause node degree statistics:** mean, variation coefficient, min, max, and entropy

Variable Graph Features:

18–21. **Node degree statistics:** mean, variation coefficient, min, and max
22–26. **Diameter:** mean, variation coefficient, min, max, and entropy

Clause Graph Features:

27–31. **Node degree statistics:** mean, variation coefficient, min, max, and entropy
32–36. **Clustering Coefficient:** mean, variation coefficient, min, max, and entropy

Balance Features:

37–41. **Ratio of positive to negative literals in each clause:** mean, variation coefficient, min, max, and entropy
42–46. **Ratio of positive to negative occurrences of each variable:** mean, variation coefficient, min, max, and entropy
47–49. **Fraction of unary, binary, and ternary clauses**

Proximity to Horn Formula:

50. **Fraction of Horn clauses**
51–55. **Number of occurrences in a Horn clause for each variable:** mean, variation coefficient, min, max, and entropy

DPLL Probing Features:

56–60. **Number of unit propagations:** computed at depths 1, 4, 16, 64 and 256
61–62. **Search space size estimate:** mean depth to contradiction, estimate of the log of number of nodes

LP-Based Features:

63–66. **Integer slack vector:** mean, variation coefficient, min, and max
67. **Ratio of integer vars in LP solution**
68. **Objective value of LP solution**

Local Search Probing Features, based on 2 seconds of running each of SAPS and GSAT:

69–78. **Number of steps to the best local minimum in a run:** mean, median, variation coefficient, 10th and 90th percentiles
79–82. **Average improvement to best in a run:** mean and coefficient of variation of improvement per step to best solution
83–86. **Fraction of improvement due to first local minimum:** mean and variation coefficient
87–90. **Coefficient of variation of the number of unsatisfied clauses in each local minimum:** mean and variation coefficient

Clause Learning Features (based on 2 seconds of running Zchaff_rand):

91–99. **Number of learned clauses:** mean, variation coefficient, min, max, 10%, 25%, 50%, 75%, and 90% quantiles
100–108. **Length of learned clauses:** mean, variation coefficient, min, max, 10%, 25%, 50%, 75%, and 90% quantiles

Survey Propagation Features

109–117. **Confidence of survey propagation:** For each variable, compute the higher of $P(true)/P(false)$ or $P(false)/P(true)$. Then compute statistics across variables: mean, variation coefficient, min, max, 10%, 25%, 50%, 75%, and 90% quantiles
118–126. **Unconstrained variables:** For each variable, compute $P(unconstrained)$. Then compute statistics across variables: mean, variation coefficient, min, max, 10%, 25%, 50%, 75%, and 90% quantiles

Timing Features

127–138. **CPU time required for feature computation:** one feature for each of 12 computational subtasks

Fig. 1. 12 groups of SAT features

and proximity to horn formula features (18–21, 50–55); diameter-based features (22–26); clause graph features (27–36); unit propagation features (56 –60); search space size estimation (61–62); LP-based features (63–68); local search probing features (69–90) with SAPS and GSAT; clause learning features (91–108); and survey propagation features (109–126).

References

1. N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75, 2005.
2. P. Herwig. Using graphs to get a better insight into satisfiability problems. Master's thesis, Delft University of Technology, Department of Electrical Engineering, Mathematics and Computer Science, 2006.
3. E. I. Hsu, C. Muise, J. C. Beck, and S. A. McIlraith. Probabilistically estimating backbones and variable bias. In *Fourteenth International Conference on Principles and Practice of Constraint Programming (CP'08)*, 2008.
4. Y. S. Mahajan, Z. Fu, and S. Malik. Zchaff2004: an efficient SAT solver. In *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 360–375, 2005.
5. E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Tenth International Conference on Principles and Practice of Constraint Programming (CP'04)*, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, 2004.
6. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *JAIR*, 32:565–606, 2008.
7. L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla2009: An automatic algorithm portfolio for SAT. Solver description, 2009 SAT Competition, 2009.