

Diversifications in Local Search for SAT

Chu Min Li¹, Yu Li¹, and Wanxia Wei

¹ MIS, Université de Picardie Jules Verne, 33 Rue St. Leu, 80039 Amiens Cedex 01, France
{chu-min.li, yu.li}@u-picardie.fr

² weiwaxia@gmail.com

Given a CNF formula \mathcal{F} and an assignment A , the objective function that local search for SAT attempts to minimize is usually the total number of unsatisfied clauses in \mathcal{F} under A . The score of a variable x with respect to A is the decrease of the objective function when it is flipped. Intensification and diversification are two strategies generally used in local search to find a satisfying assignment. The intensification strategy means to improve the objective function value as much as possible during a period of search, and the diversification strategy means to escape local minimums to exploit new regions of the search space. When intensification is relatively simple to exploit using the promising decreasing variables defined in [4], diversification is very hard to exploit.

A well known approach to diversify search is to introduce a noise p to disturb the choice of the next flipping variable [5] in a falsified clause c :

Novelty(p, c): Sort the variables in clause c by their scores, breaking ties in favor of the least recently flipped variable. Consider the best and second best variables from the sorted variables. If the best variable is not the most recently flipped one in c , then pick it. Otherwise, with probability p , pick the second best variable, and with probability $1-p$, pick the best variable.

The value of p is essential for the performance of Novelty. Unfortunately the best p is hard to obtain and is instance-specific. The adaptive noise mechanism was introduced in [3] to automatically adjust the value of p during search. We refer to this mechanism as Hoos noise mechanism. This mechanism adjusts noise based on search progress and applies the adjusted noise to variables in the chosen falsified clause in a search step.

We have proposed another adaptive noise mechanism in TNM [6]. This mechanism uses the history of the most recent consecutive falsifications of a clause. During the search, for the variables in each clause, we record both the variable that most recently falsifies this clause and the number of the most recent consecutive falsifications of this clause due to the flipping of this variable. For a clause c , we use `var_fals[c]` to denote the variable that most recently falsifies c and use `num_fals[c]` to denote the number of the most recent consecutive falsifications of c due to the flipping of this variable. Note that if c is a falsified clause, `var_fals[c]` is necessarily the most recently flipped variable in c . If it is the best variable in c , then noise is set to an appropriate value depending on `num_fals[c]` to choose the second best variable to flip. Namely, the higher `num_fals[c]` is, the higher the noise value is. This mechanism is different from Hoos noise mechanism in that it is independent of the objective function and is clause-specific.

TNM combines our clause-specific adaptive noise mechanism and Hoos noise mechanism according to the evenness of the search. A search step is said uneven in TNM if some variables are flipped much more often than others so far. The clause-specific noise is used for uneven steps and noise adapted by Hoos mechanism is used for even steps. Note that in an uneven step, The clause-specific noise is used no matter if the two best variables in the falsified clauses are often flipped or not.

We propose a new solver *adaptG2wsat2011* which also combines the clause-specific noise and Hoos noise mechanism. However, the clause-specific noise is restricted for clauses falsified much more often than other clauses. Namely, let *total_falsifications* denote the total number of clause falsifications (each time a clause is falsified, *total_falsifications* is incremented by 1) and *nb_falsification[c]* the number of falsifications of the clause *c* (each time *c* is falsified, *nb_falsification(c)* is incremented by 1). If $nb_falsification[c] > uneven_threshold * total_falsifications / nbClauses$, where *uneven_threshold* is a parameter and *nbClauses* is the number of clauses in the CNF formula, then the clause-specific noise is used to choose a variable to flip in *c*, otherwise, noise adjusted using Hoos mechanism is used. In addition the *uneven_threshold* is adjusted so that there are more even steps than uneven steps.

Note that the above noise is limited to the best two variables in *c* and other variables in *c* do not have any chance to be picked. Novelty+ [2] uses a random walk with a small probability to randomly select a variable to flip in *c*, and Novelty++ [4] flips the least recently flipped variable with a small probability. We use Novelty+ and Novelty++ respectively in even steps and uneven steps to give a chance to other variables in *c* to be flipped. In addition, in uneven case, when all variable scores in *c* are negative, we use the sparrow approach [1] to select the next variable to flip in *c*.

References

1. A. Frohlich A. Balint. Improving stochastic local search for sat with a new probability distribution. In *SAT10, LNCS 6175, Springer*, pages 10–16, 2010.
2. H. Hoos. On the run-time behavior of stochastic local search algorithms for sat. In *Proceedings of AAAI-99*, pages 661–666, 1999.
3. H. Hoos. An adaptive noise mechanism for walksat. In *Proceedings of AAAI-02*, pages 655–660. AAAI Press / The MIT Press, 2002.
4. C. M. Li and W. Q. Huang. Diversification and Determinism in Local Search for Satisfiability. In *Proceedings of SAT2005, 8th International Conference on Theory and Applications of Satisfiability Testing*, pages 158–172, 2005.
5. D.A. McAllester, B. Selman, and H. Kautz. Evidence for invariant in local search. In *Proceedings of AAAI-97*, pages 321–326, 1997.
6. C.M. Li W. Wei. Switching between two adaptive noise mechanisms in local search for sat. In *TNM solver description for the SAT 2009 competition*. <http://www.cril.univ-artois.fr/SAT09/solvers/booklet.pdf>, 2009.