Combining Adaptive Noise and Look-Ahead in Local Search for SAT[⋆,⋆⋆]

Chu Min Li[1], Wanxia Wei[2], and Harry Zhang[2]

[1] LaRIA, Université de Picardie Jules Verne
33 Rue St. Leu, 80039 Amiens Cedex 01, France
chu-min.li@u-picardie.fr

[2] Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, E3B 5A3
{wanxia.wei,hzhang}@unb.ca

Abstract. The adaptive noise mechanism was introduced in $Novelty$+ to automatically adapt noise settings during the search [4]. The local search algorithm G^2WSAT deterministically exploits promising decreasing variables to reduce randomness and consequently the dependence on noise parameters. In this paper, we first integrate the adaptive noise mechanism in G^2WSAT to obtain an algorithm $adaptG^2WSAT$, whose performance suggests that the deterministic exploitation of promising decreasing variables cooperates well with this mechanism. Then, we propose an approach that uses look-ahead for promising decreasing variables to further reinforce this cooperation. We implement this approach in $adaptG^2WSAT$, resulting in a new local search algorithm called $adaptG^2WSAT_P$. Without any manual noise or other parameter tuning, $adaptG^2WSAT_P$ shows generally good performance, compared with G^2WSAT with approximately optimal static noise settings, or is sometimes even better than G^2WSAT. In addition, $adaptG^2WSAT_P$ is favorably compared with state-of-the-art local search algorithms such as R+$adaptNovelty$+ and VW.

1 Introduction

The performance of a $Walksat$ family algorithm crucially depends on noise p and sometimes wp (random walk probability) or dp (diversification probability). For example, it is reported in [9] that running R-$Novelty$ [9] with $p = 0.4$ instead of $p = 0.6$ degrades its performance by more than 50% for random 3-SAT instances. However, to find the optimal noise settings for each heuristic, extensive experiments on various values of p and sometimes wp or dp are needed because the optimal noise settings vary widely and depend on the types and sizes of the instances.

⋆ A preliminary version of this paper was presented at the 3th International Workshop on LSCS [6], and an extended abstract of this preliminary version will appear in a book, entitled "Trends in Constraint Programming" [7].

⋆⋆ The work of the second author is partially supported by an NSERC (Natural Sciences and Engineering Research Council of Canada) PGS-D scholarship.

J. Marques-Silva and K.A. Sakallah (Eds.): SAT 2007, LNCS 4501, pp. 121–133, 2007.

To avoid manual noise tuning, two approaches were proposed. $Auto$-$Walksat$ [10] exploits the invariants observed in [9] to estimate the optimal noise settings for an algorithm on a given problem, based on several preliminary unsuccessful runs of the algorithm on this problem. This algorithm then rigorously applies the estimated optimal noise setting to the problem. The adaptive noise mechanism [4] was introduced in $Novelty$+ [3] to automatically adapt noise settings during the search, yielding the algorithm $adaptNovelty$+. This algorithm does not need any manual noise tuning and is effective for a broad range of problems.

One way to diminish the dependence of problem solving on noise settings is to reduce randomness in local search. The local search algorithm G^2WSAT deterministically selects the best promising decreasing variable to flip, if such variables exist [5]. Nevertheless, the performance of G^2WSAT still depends on static noise settings, since when there is no promising decreasing variable, a heuristic, such as $Novelty$++, is used to select a variable to flip, depending on two probabilities, p and dp. Furthermore, G^2WSAT does not favor those flips that will generate promising decreasing variables to minimize its dependence on noise settings.

In this paper, we first incorporate the adaptive noise mechanism of $adaptNovelty$+ in G^2WSAT to obtain an algorithm $adaptG^2WSAT$. Experimental results suggest that the deterministic exploitation of promising decreasing variables in $adaptG^2WSAT$ enhances this mechanism. Then, we integrate a look-ahead approach in $adaptG^2WSAT$ to favor those flips that can generate promising decreasing variables, resulting in a new local search algorithm called $adaptG^2WSAT_P$. Without any manual noise or other parameter tuning, $adaptG^2WSAT_P$ shows generally good performance, compared with G^2WSAT with approximately optimal static noise settings, or is sometimes even better than G^2WSAT. Moreover, $adaptG^2WSAT_P$ compares favorably with state-of-the-art algorithms such as R+$adaptNovelty$+ [1] and VW [11].

2 G^2WSAT and $adaptG^2WSAT$

2.1 G^2WSAT

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. Let x be a variable. The break of x, $break(x)$, is the number of clauses in $\mathcal{F}$ that are currently satisfied but will be unsatisfied if x is flipped. The make of x, $make(x)$, is the number of clauses in $\mathcal{F}$ that are currently unsatisfied but will be satisfied if x is flipped. The score of x with respect to A, $score_A(x)$, is the improvement of the objective function if x is flipped. The score of x should be the difference between $make(x)$ and $break(x)$. We write $score_A(x)$ as $score(x)$ if A is clear from the context.

Heuristics $Novelty$ [9] and $Novelty$++ [5] select a variable to flip from a randomly selected unsatisfied clause c as follows.

$Novelty(p)$: Sort the variables in 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.

$Novelty$**++**(p, dp)**:** With probability dp (diversification probability), pick the least recently flipped variable in c, and with probability 1-dp, do as $Novelty$.

Given a CNF formula $\mathcal{F}$ and an assignment A, a variable x is said to be *decreasing* with respect to A if $score_A(x) > 0$. Promising decreasing variables are defined in [5] as follows:

1. Before any flip, i.e., when A is an initial random assignment, all decreasing variables with respect to A are promising.
2. Let x and y be two different variables and x be not decreasing with respect to A. If, after y is flipped, x becomes decreasing with respect to the new assignment, then x is a promising decreasing variable with respect to the new assignment.
3. A promising decreasing variable remains promising with respect to subsequent assignments in local search until it is no longer decreasing.

G^2WSAT [5] deterministically picks the promising decreasing variable with the highest score to flip, if such variables exist. If there is no promising decreasing variable, G^2WSAT uses a heuristic, such as $Novelty$ [9], $Novelty$+ [3], or $Novelty$++ [5], to pick a variable to flip from a randomly selected unsatisfied clause.

Promising decreasing variables might be considered as the opposite of tabu variables defined in [8,9]; the flips of tabu variables are refused in a number of subsequent steps. Promising decreasing variables are chosen to flip since they probably allow local search to explore new promising regions in the search space, while tabu variables are forbidden since they probably make local search repeat or cancel earlier moves.

2.2 Algorithm $adaptG^2WSAT$

The adaptive noise mechanism [4] in $adaptNovelty$+ can be described as follows. At the beginning of a run, noise p is set to 0. Then, if no improvement in the objective function value has been observed over the last $\theta \times m$ search steps, where m is the number of the clauses of the input formula, and θ is a parameter whose default value in $adaptNovelty$+ is $1/6$, noise p is increased by $p := p + (1 - p) \times \phi$, where ϕ is another parameter whose default value in $adaptNovelty$+ is 0.2. Every time the objective function value is improved, noise p is decreased by $p := p - p \times \phi/2$.

We implement this adaptive noise mechanism of $adaptNovelty$+ in G^2WSAT to obtain an algorithm $adaptG^2WSAT$, and confirm that ϕ and θ need not be tuned for each problem instance or instance type to achieve good performances. That is, like $adaptNovelty$+, $adaptG^2WSAT$ is an algorithm in which no parameter has to be manually tuned to solve a new problem.

2.3 Performances of the Adaptive Noise Mechanism for $adaptG^2WSAT$ and for $adaptNovelty$+

We evaluate the performance of the adaptive noise mechanism for $adaptG^2WSAT$ on 9 groups of benchmark SAT problems.[1] Structured problems come from the SATLIB

[1] All experiments reported in this paper are conducted in Chorus, which consists of 2 dual processor master nodes (Sun V65) with hyperthreading enabled and 80 dual processor compute nodes (Sun V60). Each compute node has two 2.8GHz Intel Xeon processors with 2 to 3 Gigabytes of memory.

repository[2] and Miroslav Velev's SAT Benchmarks.[3] These structured problems include bw_large.c and bw_large.d in Blocksworld, 3bit*31, 3bit*32, e0ddr2*1, e0ddr2*4, enddr2*1, enddr2*8, ewddr2*1, and ewddr2*8 in Beijing, the first 5 instances in Flat200-479, logistics.c and logistics.d in logistics, par16-1, par16-2, par16-3, par16-4, and par16-5 in parity, the 10 satisfiable instances in QG, and all satisfiable formulas in Superscalar Suite 1.0a (SSS.1.0a) except for *bug54.[4] Since these 10 QG instances contain unit clauses, we simplify them using *my_compact*[5] before running every algorithm. Random problems consist of unif04-52, unif04-62, unif04-65, unif04-80, unif04-83, unif04-86, unif04-91, and unif04-99, from the random category in the SAT 2004 competition benchmark.[6] Industrial problems comprise v*1912, v*1915, v*1923, v*1924, v*1944, v*1955, v*1956, and v*1959, from the industrial category in the SAT 2005 competition benchmark.[7]

Table 1 shows the performances of $adaptG^2WSAT$ and G^2WSAT, both using heuristic $Novelty$+, compared with those of $adaptNovelty$+ and $Novelty$+. This table presents the results of these algorithms for only one instance from each group. The random walk probability (wp) is not adjusted and takes the default value 0.01 for the original $Novelty$+, in each algorithm for each instance. G^2WSAT (version 2005) is downloaded from http://www.laria.u-picardie.fr/˜cli. $Novelty$+ and $adaptNovelty$+

Table 1. Performance of the adaptive noise mechanism for $adaptG^2WSAT$ using $Novelty+$ and for $adaptNovelty$+. Results in bold indicate the lower degradation in success rate.

algorithm heuristic parameters	cutoff	$Novelty+$ wp=0.01		$adaptNovelty$+ $\theta=1/6,\phi=0.2$		G^2WSAT $Novelty+$ wp=0.01		$adaptG^2WSAT$ $Novelty+$ $\theta=1/6,\phi=0.2$	
		p	suc	suc	suc degr	p	suc	suc	suc degr
bw_large.d	10^8	.17	100%	92.80%	7.20%	.20	100%	100%	**0%**
ewddr2*8	10^7	.78	100%	5.20%	94.80%	.52	100%	100%	**0%**
flat200-5	10^8	.54	99.60%	99.20%	0.40%	.60	100%	100%	**0%**
logistics.c	10^5	.41	58.00%	43.20%	25.52%	.52	81.20%	73.20%	**9.85%**
par16-1	10^9	.80	98.00%	42.80%	56.33%	.63	100%	100%	**0%**
qg5-11	10^6	.29	100%	97.20%	**2.80%**	.32	100%	92.40%	7.60%
*bug17	10^7	.82	100%	32.80%	67.20%	.29	66.00%	66.00%	**0%**
unif04-52	10^8	.51	99.60%	94.40%	5.22%	.52	100%	99.20%	**0.80%**
v*1912	10^7	.16	56.00%	50.80%	9.29%	.22	84.00%	81.20%	**3.33%**

are from $UBCSAT$ [13]. The static noise p of G^2WSAT is approximately optimal for G^2WSAT on each instance, and is obtained by comparing p = 0.10, 0.11, ..., 0.89, and 0.90 for each instance. The static noise p of $Novelty$+ is different from that of G^2WSAT because $Novelty$+ with its own noise p can perform better than

[2] http://www.satlib.org/

[3] http://www.ece.cmu.edu/∼mvelev/sat_benchmarks.html

[4] The instance *bug54 is hard for every algorithm discussed in this paper.

[5] available at http://www.laria.u-picardie.fr/˜cli

[6] http://www.lri.fr/∼simon/contest04/results/

[7] http://www.lri.fr/∼simon/contest/results/

$Novelty$+ with the noise p of G^2WSAT. Each instance is executed 250 times. The success rate of an algorithm for an instance is the number of successful runs divided by 250, and the success rate is intended to be the empirical probability with which the algorithm finds a solution for the instance within the cutoff. For each algorithm on each instance, we report the cutoff ("cutoff") and success rate ("suc"). Let sr be the success rate of G^2WSAT or $Novelty$+ with static noise for an instance, and ar the success rate of $adaptG^2WSAT$ or $adaptNovelty$+ for the same instance. For each instance, we also report the degradation ("suc degr") in success rate of $adaptG^2WSAT$, ((sr-ar)/sr)*100, compared with that of G^2WSAT, and the degradation ("suc degr") in success rate of $adaptNovelty$+, ((sr-ar)/sr)*100, compared with that of $Novelty$+.

According to Table 1, without manual noise tuning, $adaptG^2WSAT$ and $adaptNovelty$+, with the adaptive noise mechanism, achieve good performances, θ and ϕ taking the same fixed values for all problems. Nevertheless, with instance specific noise settings, G^2WSAT and $Novelty$+ achieve success rates the same as or higher than $adaptG^2WSAT$ and $adaptNovelty$+, respectively, for all instances. For all instances except for qg5-11, the degradation in success rate of $adaptG^2WSAT$ compared with that of G^2WSAT is lower than the degradation in success rate of $adaptNovelty$+ compared with that of $Novelty$+. Especially, for bw_large.d, ewddr2*8, par16-1, and *bug17, the degradation in success rate of $adaptG^2WSAT$ compared with that of G^2WSAT is significantly lower than the degradation in success rate of $adaptNovelty$+ compared with that of $Novelty$+.

In Table 1, both $adaptG^2WSAT$ and G^2WSAT use $Novelty$+ to select a variable to flip when there is no promising decreasing variable. Furthermore, $adaptG^2WSAT$ uses the same default values for parameters θ and ϕ as $adaptNovelty$+, to adapt noise. So, it appears that, apart from the implementation details, the only difference between G^2WSAT and $Novelty$+, and between $adaptG^2WSAT$ and $adaptNovelty$+, in Table 1, is the deterministic exploitation of promising decreasing variables in G^2WSAT and $adaptG^2WSAT$. From this table, we observe that the degradation in performance of $adaptG^2WSAT$ compared with that of G^2WSAT is lower than the degradation in performance of $adaptNovelty$+ compared with that of $Novelty$+. This observation suggests that the deterministic exploitation of promising decreasing variables enhances the adaptive noise mechanism. We then expect that better exploitation of promising decreasing variables will further enhance this mechanism.

3 Look-Ahead for Promising Decreasing Variables

3.1 Promising Score of a Variable

Given a CNF formula $\mathcal{F}$ and an assignment A, let x be a variable, let B be obtained from A by flipping x, and let x' be the best promising decreasing variable with respect to B. We define the promising score of x with respect to A as

$$pscore_A(x) = score_A(x) + score_B(x')$$

where $score_A(x)$ is the score of x with respect to A and $score_B(x')$ is the score of x' with respect to B.[8]

[8] x' has the highest $score_B(x')$ among all promising decreasing variables with respect to B.

If there are promising decreasing variables with respect to B, the promising score of x with respect to A represents the improvement in the number of unsatisfied clauses under A by flipping x and then x'. In this case, $pscore_A(x) > score_A(x)$.

If there is no promising decreasing variable with respect to B,

$$pscore_A(x) = score_A(x)$$

since $adaptG^2WSAT$ does not know in advance which variable will be flipped for B (the choice of the variable to flip is made randomly by using $Novelty{+}{+}$).

Given $\mathcal{F}$ and two variables x and y in $\mathcal{F}$, y is said to be a neighbor of x with respect to $\mathcal{F}$ if y occurs in some clause containing x in $\mathcal{F}$. According to Equation 6 in [5], the flipping of x can only change the scores of the neighbors of x. Given an initial assignment, G^2WSAT or $adaptG^2WSAT$ computes the scores for all variables, and then uses Equation 6 in [5] to update the scores of the neighbors of the flipped variable after each step and maintains a list of promising decreasing variables. This update takes time $O(L)$, where L is the upper bound for the sum of the lengths of all clauses containing the flipped variable and is almost a constant for a random 3-SAT problem when the ratio of the number of clauses to the number of variables is a constant. The computation of $pscore_A(x)$ involves the simulation of flipping x and the searching for the largest score of the promising decreasing variables after flipping x. This computation takes time $O(L + \gamma)$, where γ is the upper bound for the number of all the promising decreasing variables in $\mathcal{F}$ after flipping x.

3.2 Integrating Limited Look-Ahead in $adaptG^2WSAT$

We improve $adaptG^2WSAT$ in two ways. The algorithm $adaptG^2WSAT$ maintains a stack, $DecVar$, to store all promising decreasing variables in each step. When there are promising decreasing variables, the improved $adaptG^2WSAT$ chooses the least recently flipped promising decreasing variable among all promising decreasing variables

Function: $Novelty{+}_P$(p, wp, c)

with probability wp **do** $y \leftarrow$ randomly choose a variable in c;
otherwise
  Determine $best$ and $second$, breaking ties in favor of the least recently flipped variable;
  /*$best$ and $second$ are the best and second best variables in c according to the scores*/
  if $best$ is the most recently flipped variable in c
  then
    with probability p **do** $y \leftarrow second$;
    otherwise if $pscore(second) >= pscore(best)$ **then** $y \leftarrow second$ **else** $y \leftarrow best$;
  else
    if $best$ is more recently flipped than $second$
    then if $pscore(second) >= pscore(best)$ **then** $y \leftarrow second$ **else** $y \leftarrow best$;
    else $y \leftarrow best$;
return y;

Fig. 1. Function $Novelty{+}_P$

in $|DecVar|$ to flip. Otherwise, the improved $adaptG^2WSAT$ selects a variable to flip from a randomly chosen unsatisfied clause c, using heuristic $Novelty$+$_P$ (see Fig. 1), which extends $Novelty$+ [3], to exploit limited look-ahead.

Let $best$ and $second$ denote the best and second best variables respectively, measured by the scores of variables in c. $Novelty$+$_P$ computes the promising scores for only $best$ and $second$, only when $best$ is more recently flipped than $second$ (including the case in which $best$ is the most recently flipped variable, where the computation is performed with probability $1 - p$), in order to favor the less recently flipped $second$. In this case, $score(second) < score(best)$. As is suggested by the success of $HSAT$ [2] and $Novelty$ [9], a less recently flipped variable is generally better if it can improve the objective function at least as well as a more recently flipped variable does. Accordingly, $Novelty$+$_P$ prefers $second$ if $second$ is less recently flipped than $best$ and if $pscore(second) \geq pscore(best)$.

The improved $adaptG^2WSAT$ is called $adaptG^2WSAT_P$ and is sketched in Fig. 2. Note that wp (random walk probability) is also automatically adjusted and $wp = p/10$. The reason for adjusting wp this way is that, when noise needs to be high, local search should also be well randomized, and when low noise is sufficient, random walks are often not needed. The setting $wp = p/10$ comes from the fact that $p = 0.5$ and $dp = 0.05$ give the best results for random 3-SAT instances in G^2WSAT.

Given a CNF formula $\mathcal{F}$ and an assignment A, the set of assignments obtained by flipping one variable of $\mathcal{F}$ is called the *1-flip neighborhood* of A, and the set of assignments obtained by flipping two variables of $\mathcal{F}$ is called the *2-flip neighborhood* of

---

Algorithm. $adaptG^2WSAT_P$(SAT-formula $\mathcal{F}$)

---

for try=1 **to** *Maxtries* **do**
    $A \leftarrow$ randomly generated truth assignment; p=0; wp=0;
    Store all decreasing variables in stack *DecVar*;
    for *flip*=1 **to** *Maxsteps* **do**
        if A satisfies $\mathcal{F}$ **then return** A;
        if $|DecVar| > 0$
        then
            $y \leftarrow$the least recently flipped promising decreasing variable among
                all promising decreasing variables in $|DecVar|$;
        else
            $c \leftarrow$randomly selected unsatisfied clause under A;
            $y \leftarrow Novelty$+$_P$(p, wp, c);
        $A \leftarrow A$ with y flipped;
        Adapt p and wp;
        Delete variables that are no longer decreasing from *DecVar*;
        Push new decreasing variables into *DecVar* which are different from
         y and were not decreasing before y is flipped;
return Solution not found;

Fig. 2. Algorithm $adaptG^2WSAT_P$

A. The algorithm $adaptG^2WSAT_P$ exploits only the 1-flip neighborhoods, since the limited look-ahead is just used as a heuristic to select the next variable to flip.

We find that in $adaptG^2WSAT$ and $adaptG^2WSAT_P$, which use heuristics $Novelty$++ and $Novelty$+$_P$, respectively, $\theta = 1/5$ and $\phi = 0.1$ give slightly better results on the 9 groups of instances presented in Section 2.3 than $\theta = 1/6$ and $\phi = 0.2$, their original default values in $adaptNovelty$+. So, in $adaptG^2WSAT_P$, $\theta = 1/5$ and $\phi = 0.1$.

In this paper, $adaptG^2WSAT_P$ is improved in two ways, based on the preliminary $adaptG^2WSAT_P$ described in the preliminary version of this paper [6,7]. The first improvement is that, when promising decreasing variables exist, $adaptG^2WSAT_P$ no longer computes the promising scores for the δ promising decreasing variables with higher scores in $|DecVar|$, where δ is a parameter, but chooses the least recently flipped promising decreasing variable among all promising decreasing variables in $|DecVar|$ to flip. As a result, $adaptG^2WSAT_P$ no longer needs parameter δ. The reasons for this first improvement are that, usually the scores of promising decreasing variables are close and so such variables can improve the objective function roughly the same, and that flipping the least recently flipped promising decreasing variable can increase the mobility and coverage [12] of a local search algorithm in the search space. The second improvement is that, when there is no promising decreasing variable, $adaptG^2WSAT_P$ uses $Novelty$+$_P$ instead of $Novelty$++$_P$ [6,7], to select a variable to flip from a randomly chosen unsatisfied clause c. The difference between $Novelty$+$_P$ and $Novelty$++$_P$ is that, with wp (random walk probability), $Novelty$+$_P$ randomly chooses a variable to flip from c, but with dp (diversification probability), $Novelty$++$_P$ chooses a variable in c, whose flip will falsify the least recently satisfied clause. Considering that $adaptG^2WSAT_P$ deterministically uses both promising decreasing variables and promising scores, adding a small amount of randomness[9] to the search may help find a solution.

4 Evaluation

We evaluate $adaptG^2WSAT_P$ on the 9 groups of instances, or the 56 instances, presented in Section 2.3. For an instance and an algorithm, we report the median flip number ("#flips") and the median run time ("time") in seconds, for this algorithm to find a solution for this instance. Each instance is executed 250 times. If an algorithm can successfully find a solution for an instance in at least 126 runs, the median flip number and median run time are calculated based on these 250 runs. If an algorithm cannot achieve a success rate greater than 50% on an instance even if the cutoff is greater than or equal to the maximum value among the cutoffs of all other algorithms, the median flip number and median run time cannot be calculated; we use "$> Maxsteps$" (greater than $Maxsteps$) to denote the median flip number and use "n/a" to denote the median run time, where $Maxsteps$ is the cutoff for this algorithm on this instance. If the median flip number and median run time of G^2WSAT with any noise settings for an instance cannot be calculated, we also use n/a to denote the optimal noise setting. Results in bold indicate the best performance for an instance.

[9] In general, wp ranges from 0% to 10%.

4.1 Comparison of Performances of $adaptG^2WSAT_P$, G^2WSAT, and $adaptG^2WSAT$

We compare the performances of $adaptG^2WSAT_P$, G^2WSAT with approximately optimal noise settings, and $adaptG^2WSAT$ in Table 2, where $adaptG^2WSAT_P$ uses $Novelty$+$_P$, and G^2WSAT and $adaptG^2WSAT$ use $Novelty$++, to pick a variable to flip, when there is no promising decreasing variable. On the instances that G^2WSAT can solve in reasonable time, except for qg7-13, the performance of $adaptG^2WSAT_P$ is comparable to that of G^2WSAT with approximately optimal noise settings. Moreover, $adaptG^2WSAT_P$ can solve 3bit*31, 3bit*32, *bug5, *bug38, *bug39, and *bug40, which are hard for G^2WSAT with any static noise settings. More importantly, $adaptG^2WSAT_P$ does not need any manual tuning of p and wp for each instance while G^2WSAT needs manual tuning of p and dp for each instance. In other words, G^2WSAT cannot achieve the performance shown in this table by using the same p and dp for the broad range of instances.

On the instances that $adaptG^2WSAT$ can solve in reasonable time, the performance of $adaptG^2WSAT_P$ is comparable to that of $adaptG^2WSAT$. Furthermore, $adaptG^2WSAT_P$ can solve 3bit*31, 3bit*32, *bug5, *bug38, *bug39, and *bug40, which are hard for $adaptG^2WSAT$. In addition, among the 56 instances presented in this table, $adaptG^2WSAT_P$ exhibits the best run time performance and/or the best flip number performance on the 13 instances among $adaptG^2WSAT_P$, G^2WSAT with approximately optimal noise settings, and $adaptG^2WSAT$, while $adaptG^2WSAT$ is never the best.

4.2 Comparison of Performances of $adaptG^2WSAT_P$, R+$adaptNovelty$+, and VW

R+$adaptNovelty$+ is $adaptNovelty$+ with preprocessing to add a set of resolvents of length ≤ 3 into the input formula [1]. VW [11] is an extension of $Walksat$. VW adjusts and smoothes variable weights, and takes variable weights into account when selecting a variable to flip. R+$adaptNovelty$+, G^2WSAT with p=0.50 and dp=0.05, and VW won the gold, silver, and bronze medals, respectively, in the satisfiable random formula category in the SAT 2005 competition.[10]

Table 3 compares the performance of $adaptG^2WSAT_P$ with the performances of R+$adaptNovelty$+ and VW. We download R+$adaptNovelty$+ and VW from http://www.satcompetition.org/. We use the default value 0.01 for the random walk probability in R+$adaptNovelty$+, when running this algorithm. In this table, instances with † on the right constitute the entire set of instances that were used to originally evaluate R+$adaptNovelty$+ in [1]. Among the 56 instances presented in this table, in terms of run time, $adaptG^2WSAT_P$, R+$adaptNovelty$+, and VW are the best algorithms on the 32, 16, and 13 instances, respectively. Also, among the 56 instances, in terms of run time, $adaptG^2WSAT_P$ outperforms R+$adaptNovelty$+ and VW on the 38 and 42 instances, respectively.

[10] http://www.satcompetition.org/

Table 2. Performance of $adaptG^2WSAT_P$, $adaptG^2WSAT$, and G^2WSAT with approximately optimal noise settings

	$adaptG^2WSAT_P$		$adaptG^2WSAT$		G^2WSAT		
	#flips	time	#flips	time	optimal	#flips	time
bw_large.c	**1083947**	**3.650**	3553694	10.175	(.21, 0)	2119497	3.699
bw_large.d	**1542898**	8.590	9626411	49.635	(.16, 0)	3237895	**7.180**
3bit*31	**87158**	**0.780**	$> 10^7$	n/a	n/a	$> 10^7$	n/a
3bit*32	**60518**	**0.565**	$> 10^7$	n/a	n/a	$> 10^7$	n/a
e0ddr2*1	4520164	19.275	831073	2.595	(.14, .09)	**254182**	**0.910**
e0ddr2*4	641587	2.855	208815	0.805	(.23, .1)	**117266**	**0.540**
enddr2*1	982540	4.570	153905	0.640	(.18, .1)	**97451**	**0.535**
enddr2*8	412624	2.385	135332	0.585	(.16, .09)	**90076**	**0.480**
ewddr2*1	492907	2.470	137430	0.600	(.18, .1)	**89420**	**0.505**
ewddr2*8	262177	1.385	116917	0.535	(.16, .1)	**67854**	**0.425**
flat200-1	36764	0.025	42053	0.020	(.49, .08)	**25358**	**0.010**
flat200-2	288521	0.160	303515	0.135	(.49, .07)	**171487**	**0.085**
flat200-3	71324	0.045	89515	0.040	(.51, .05)	**51037**	**0.025**
flat200-4	314273	0.180	323353	0.145	(.49, .05)	**178842**	**0.095**
flat200-5	4963846	2.675	4173580	1.810	(.49, .08)	**3008035**	**1.455**
logistics.c	54777	0.075	46875	0.060	(.24, .07)	**38177**	**0.040**
logistics.d	83894	0.185	102575	0.165	(.2, .08)	**78013**	**0.105**
par16-1	58937999	27.955	76985828	29.870	(.51, .01)	**48342381**	**20.835**
par16-2	130634181	64.300	140615726	57.170	(.59, .01)	**73324801**	**32.460**
par16-3	104764223	50.865	112297525	44.885	(.58, .01)	**80700698**	**33.223**
par16-4	133899858	63.595	174053106	68.735	(.5, .02)	**89662042**	**39.256**
par16-5	124873168	59.865	133250726	53.385	(.54, .02)	**83818097**	**35.688**
qg1-07	6413	0.025	7370	0.020	(.38, 0)	**4599**	**0.010**
qg1-08	361229	4.740	448660	3.635	(.11, .03)	**339312**	**1.350**
qg2-07	3869	0.020	4708	0.025	(.33, .01)	**2648**	**0.005**
qg2-08	**1262398**	8.960	1473258	9.565	(.22, 0)	1449931	**6.270**
qg3-08	36322	0.125	36046	0.040	(.44, .05)	**20517**	**0.015**
qg4-09	68472	0.310	70659	0.100	(.37, 0)	**48741**	**0.075**
qg5-11	20598	0.210	23431	0.275	(.38, .01)	**12559**	**0.080**
qg6-09	414	0.005	441	0.005	(.41, .08)	**340**	**0.000**
qg7-09	392	0.005	318	0.005	(.41, .1)	**316**	**0.015**
qg7-13	$> 10^8$	n/a	$> 10^8$	n/a	(.33, 0)	**4768987**	**50.809**
*bug3	$> 10^8$	n/a	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug4	$> 10^8$	n/a	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug5	**1460519**	**6.050**	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug17	107501	**1.170**	425730	5.130	(.15, .15)	**63582**	1.355
*bug38	**181666**	**0.745**	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug39	**75743**	**0.390**	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug40	**182279**	**0.890**	$> 10^8$	n/a	n/a	$> 10^8$	n/a
*bug59	102853	1.080	268332	2.475	(.62, .06)	**52276**	**0.408**
unif04-52	5588325	6.065	6763462	5.570	(.4, .07)	**4991465**	**4.295**
unif04-62	530432	0.590	768215	0.640	(.49, .03)	**386031**	**0.335**
unif04-65	1406786	1.560	1566427	1.315	(.48, .06)	**1289658**	**0.918**
unif04-80	3059121	3.575	3751125	3.300	(.45, .1)	**1908125**	**1.760**
unif04-83	8370126	9.930	6589739	5.860	(.43, .09)	**4370302**	**3.112**
unif04-86	6288398	7.450	5817258	5.250	(.43, .09)	**3429233**	**2.442**
unif04-91	659313	0.780	789717	0.730	(.5, .05)	**414399**	**0.324**
unif04-99	**4054201**	4.985	7746102	7.205	(.45, .02)	4931360	**4.530**
v*1912	**3454184**	84.115	3683237	78.625	(.16, 0)	3554771	**65.509**
v*1915	12928287	409.480	14636382	328.450	(.19, .02)	**12510065**	**288.966**
v*1923	1200896	25.030	1358055	16.630	(.42, 0)	**1065848**	**13.386**
v*1924	**1389813**	28.040	1756779	29.855	(.21, .04)	1613496	**23.019**
v*1944	4248279	216.700	4386535	156.67	(.20, 0)	**3667138**	**126.398**
v*1955	1404357	56.240	1417356	32.195	(.29, .01)	**1152386**	**28.669**
v*1956	1762589	71.100	1849539	68.365	(.26, .02)	**1599232**	**46.434**
v*1959	612589	27.985	786925	32.815	(.37, .01)	**498563**	**16.276**

Table 3. Experimental results for R+$adaptNovelty$+, $adaptG^2WSAT_P$, and VW

	R+$adaptNovelty$+		$adaptG^2WSAT_P$		VW	
	#flips	time	#flips	time	#flips	time
bw_large.c†	9489817	29.140	**1083947**	**3.650**	1868393	5.960
bw_large.d	27179763	152.160	**1542898**	**8.590**	2963500	18.120
3bit*31	152565	1.645	87158	0.780	**37487**	**0.290**
3bit*32	133945	1.640	60518	0.565	**21858**	**0.160**
e0ddr2*1†	**2488226**	**10.630**	4520164	19.275	6549282	22.530
e0ddr2*4†	**355044**	**1.530**	641587	2.855	1894243	7.850
enddr2*1†	**331420**	**1.555**	982540	4.570	4484178	17.605
enddr2*8†	**11753**	**0.020**	412624	2.385	3493986	15.505
ewddr2*1†	**154825**	**0.675**	492907	2.470	4714786	18.410
ewddr2*8†	**32527**	**0.100**	262177	1.385	4956356	21.785
flat200-1	50600	0.030	**36764**	**0.025**	187053	0.085
flat200-2	535300	0.280	**288521**	**0.160**	1318485	0.650
flat200-3	161169	0.085	**71324**	**0.045**	664550	0.330
flat200-4	577180	0.290	**314273**	**0.180**	2747696	1.345
flat200-5	15841761	8.366	**4963846**	**2.675**	26137279	13.119
logistics.c†	57693	**0.075**	**54777**	**0.075**	70446	0.085
logistics.d	162737	0.220	**83894**	**0.185**	340379	0.395
par16-1†	80339283	37.645	**58937999**	**27.955**	$> 10^9$	n/a
par16-2†	324826713	157.455	**130634181**	**64.300**	$> 10^9$	n/a
par16-3†	224140856	107.410	**104764223**	**50.865**	$> 10^9$	n/a
par16-4†	274054172	129.660	**133899858**	**63.595**	$> 10^9$	n/a
par16-5†	264871971	125.025	**124873168**	**59.865**	$> 10^9$	n/a
qg1-07†	9882	**0.015**	**6413**	0.025	21304	0.055
qg1-08†	676122	**2.300**	**361229**	4.740	2548200	69.325
qg2-07†	6147	**0.010**	**3869**	0.020	9181	0.035
qg2-08†	2200276	**8.440**	**1262398**	8.960	8843525	277.735
qg3-08†	53998	**0.070**	**36322**	0.125	137354	0.185
qg4-09†	105386	**0.165**	**68472**	0.310	264297	0.505
qg5-11†	36856	0.215	**20598**	**0.210**	39907	0.410
qg6-09†	542	**0.000**	**414**	**0.000**	1014	**0.000**
qg7-09†	531	**0.000**	**392**	**0.000**	1037	**0.000**
qg7-13†	**5113772**	**66.680**	$> 10^8$	n/a	8843466	307.620
*bug3	62148492	360.920	$> 10^8$	n/a	**1974994**	**4.875**
*bug4	$> 10^8$	n/a	$> 10^8$	n/a	**177511**	**0.460**
*bug5	66283256	431.395	1460519	6.050	**280071**	**0.735**
*bug17	6020734	141.875	107501	1.170	**32999**	**0.275**
*bug38	4699436	32.735	181666	0.745	**157834**	**0.385**
*bug39	9693455	54.345	**75743**	0.390	83287	**0.220**
*bug40	17465338	125.010	182279	0.890	**98834**	**0.290**
*bug59	389865	4.150	102853	1.080	**66090**	**0.345**
unif04-52†	24720067	21.335	**5588325**	**6.065**	22594215	17.115
unif04-62†	1484946	1.280	**530432**	**0.590**	3321105	2.605
unif04-65†	9043996	7.885	**1406786**	**1.560**	4505318	3.520
unif04-80†	5432957	4.780	**3059121**	**3.575**	20083928	16.515
unif04-83†	291310536	255.685	**8370126**	**9.930**	25897048	21.590
unif04-86†	38667651	34.045	**6288398**	7.450	8536496	**7.170**
unif04-91†	1581843	1.370	**659313**	**0.780**	3097695	2.725
unif04-99†	16856278	14.850	**4054201**	**4.985**	17422353	15.400
v*1912	6812718	148.735	**3454184**	**84.115**	61152892	3037.695
v*1915	78909897	2208.900	**12928287**	**409.480**	$> 10^8$	n/a
v*1923	2736569	51.662	**1200896**	**25.030**	9820793	340.430
v*1924	2931225	60.319	**1389813**	**28.040**	13744232	515.720
v*1944	6153990	373.905	**4248279**	**216.700**	58541545	7971.731
v*1955	2755333	89.455	**1404357**	**56.240**	10396220	1073.960
v*1956	2865074	114.685	**1762589**	**71.100**	13419375	1437.035
v*1959	2420412	118.335	**612589**	**27.985**	11433482	1377.245

Table 4. Experimental results for the preliminary $adaptG^2WSAT_P$ and $adaptG^2WSAT_P$

	preliminary $adaptG^2WSAT_P$		$adaptG^2WSAT_P$	
	#flips	time	#flips	time
*bug5	$> 10^8$	n/a	**1460519**	**6.050**
*bug17	133691	2.820	**107501**	**1.170**
*bug38	$> 10^8$	n/a	**181666**	**0.745**
*bug39	$> 10^8$	n/a	**75743**	**0.390**
*bug40	$> 10^8$	n/a	**182279**	**0.890**
*bug59	179091	4.965	**102853**	**1.080**

4.3 Comparison of Performances of $adaptG^2WSAT_P$ and Preliminary $adaptG^2WSAT_P$

Our experimental results show that $adaptG^2WSAT_P$ exhibits better performance than the preliminary $adaptG^2WSAT_P$ on some instances from SSS.1.0a presented in Section 2.3. According to our experimental results, on the remaining instances presented in Section 2.3, the overall performance of $adaptG^2WSAT_P$ is close to that of the preliminary $adaptG^2WSAT_P$. Table 4 indicates that $adaptG^2WSAT_P$ exhibits good performance on the 6 instances from SSS.1.0a while the preliminary $adaptG^2WSAT_P$ has difficulty on 4 out of these 6.

5 Conclusion

We have found that the deterministic exploitation of promising decreasing variables can enhance the adaptive noise mechanism in local search for SAT, and thus integrated this adaptive noise mechanism in G^2WSAT to obtain the algorithm $adaptG^2WSAT$. We then have proposed a limited look-ahead approach to favor those flips generating promising decreasing variables to further improve the adaptive noise mechanism. The look-ahead approach is based on the promising scores of variables, meaning that after flipping a variable x, the score of the best promising decreasing variable should be added to the score of x to improve the objective function. The resulting algorithm is called $adaptG^2WSAT_P$.

There are two new parameters in $adaptG^2WSAT_P$, θ and ϕ, which are from $adaptNovelty$+ and are used to implement the adaptive noise mechanism. However, noise p and random walk probability wp are entirely automatically adapted. Our experimental results confirm that, like θ and ϕ in $adaptNovelty$+, θ and ϕ in $adaptG^2WSAT_P$ are substantially less sensitive to problem instances and problem types than are p and wp [4], and our results also show that the same fixed default values of θ and ϕ allow $adaptG^2WSAT_P$ to achieve good performances for a broad range of SAT problems. Moreover, our experimental results show that, without any manual noise or other parameter tuning, $adaptG^2WSAT_P$ shows generally good performance, compared with G^2WSAT with approximately optimal static noise settings, or is sometimes even better than G^2WSAT, and that $adaptG^2WSAT_P$ compares favorably with state-of-the-art algorithms such as R+$adaptNovelty$+ and VW.

We plan to optimize the computation of promising scores, which actually is not incremental. In addition, the efficient implementation techniques of $UBCSAT$, the variable weight smoothing technique proposed in VW, and the preprocessing used in R+$adaptNovelty$+ could be integrated into $adaptG^2WSAT_P$.

References

1. Anbulagan, D. N. Pham, J. Slaney, and A. Sattar. Old Resolution Meets Modern SLS. In *Proceedings of AAAI-2005*, pages 354–359. AAAI Press, 2005.
2. I. P. Gent and T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. In *Proceedings of AAAI-1993*, pages 28–33. AAAI Press, 1993.
3. H. Hoos. On the Run-Time Behavior of Stochastic Local Search Algorithms for SAT. In *Proceedings of AAAI-1999*, pages 661–666. AAAI Press, 1999.
4. H. Hoos. An Adaptive Noise Mechanism for WalkSAT. In *Proceedings of AAAI-2002*, pages 655–660. AAAI Press, 2002.
5. C. M. Li and W. Q. Huang. Diversification and Determinism in Local Search for Satisfiability. In *Proceedings of SAT-2005*, pages 158–172. Springer, LNCS 3569, 2005.
6. C. M. Li, W. Wei, and H. Zhang. Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In *Proceedings of LSCS-2006*, pages 2–16, 2006.
7. C. M. Li, W. Wei, and H. Zhang. Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In F. Benhamou, N. Jussien, and B. O'Sullivan, editors, *Trends in Constraint Programming*, chapter 2. Hermes Science, 2007 (to appear).
8. B. Mazure, L. Sais, and E. Gregoire. Tabu Search for SAT. In *Proceedings of AAAI-1997*, pages 281–285. AAAI Press, 1997.
9. D. A. McAllester, B. Selman, and H. Kautz. Evidence for Invariant in Local Search. In *Proceedings of AAAI-1997*, pages 321–326. AAAI Press, 1997.
10. D. J. Patterson and H. Kautz. Auto-Walksat: A Self-Tuning Implementation of Walksat. Electronic Notes on Discrete Mathematics 9, 2001.
11. S. Prestwich. Random Walk with Continuously Smoothed Variable Weights. In *Proceedings of SAT-2005*, pages 203–215. Springer, LNCS 3569, 2005.
12. D. Schuurmans and F. Southey. Local Search Characteristics of Incomplete SAT Procedures. In *Proceedings of AAAI-2000*, pages 297–302. AAAI Press, 2000.
13. D. A. D. Tompkins and H. H. Hoos. UBCSAT: An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT. In *Proceedings of SAT-2004*, pages 306–315. Springer, LNCS 3542, 2004.