Switching among Non-Weighting, Clause Weighting, and Variable Weighting in Local Search for SAT*

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

[1] Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, E3B 5A3
{wanxia.wei,hzhang}@unb.ca
[2] MIS, Université de Picardie Jules Verne 33 Rue St. Leu, 80039 Amiens Cedex 01, France
chu-min.li@u-picardie.fr

Abstract. One way to design a local search algorithm that is effective on many types of instances is allowing this algorithm to switch among heuristics. In this paper, we refer to the way in which non-weighting algorithm $adaptG^2WSAT+$ selects a variable to flip, as *heuristic* $adaptG^2WSAT+$, the way in which clause weighting algorithm $RSAPS$ selects a variable to flip, as *heuristic* $RSAPS$, and the way in which variable weighting algorithm VW selects a variable to flip, as *heuristic* VW. We propose a new switching criterion: the evenness or unevenness of the distribution of clause weights. We apply this criterion, along with another switching criterion previously proposed, to *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW. The resulting local search algorithm, which adaptively switches among these three heuristics in every search step according to these two criteria to intensify or diversify the search when necessary, is called $NCVW$ (Non-, Clause, and Variable Weighting). Experimental results show that $NCVW$ is generally effective on a wide range of instances while $adaptG^2WSAT+$, $RSAPS$, VW, and $gNovelty+$ and $adaptG^2WSAT0$, which won the gold and silver medals in the satisfiable random category in the SAT 2007 competition, respectively, are not.

1 Introduction

Intensification refers to search strategies that intend to greedily improve solution quality or the chances of finding a solution in the near future [5]. Diversification refers to search strategies that help achieve a reasonable coverage when exploring the search space in order to avoid search stagnation and entrapment in relatively confined regions of the search space that may contain only locally optimal solutions [5]. Generally speaking, there are three classes of local search algorithm: non-weighting, clause weighting, and variable weighting. A non-weighting algorithm does not use any weighting and mainly focuses on intensifying the search to greatly decrease the number of unsatisfied clauses. A clause weighting algorithm uses clause weighting to diversify the search while a variable weighting algorithm uses variable weighting to diversify the search.

* The work of the first author is partially supported by NSERC PGS-D (Natural Sciences and Engineering Research Council of Canada Post-Graduate Scholarships for Doctoral students).

P.J. Stuckey (Ed.): CP 2008, LNCS 5202, pp. 313–326, 2008.

Efforts have been made to develop non-weighting local search algorithms [3,8,9,10,14,15]. Among these algorithms, $adaptG^2WSAT_P$ [9], the improved $adaptG^2WSAT_P$ [10], $adaptG^2WSAT0$ [8], and $adaptG^2WSAT+$ [15] combine the use of promising decreasing variables defined in [7] with the adaptive noise mechanism proposed in [4]. According to the definition of promising decreasing variables, flipping such variables not only decreases the number of unsatisfied clauses but also probably allows local search to explore new promising regions in the search space.

Clause weighting has been used in local search algorithms to help the search escape from local minima and diversify the search [6,11,12]. These algorithms include $Breakout$ [11], $SAPS$ (Scaling And Probabilistic Smoothing) [6], $RSAPS$ (Reactive SAPS) [6], and $gNovelty+$ [12]. Of these algorithms, $gNovelty+$ combines the use of promising decreasing variables and clause weighting techniques. Rather than using clause weighting, the local search algorithm VW [13] employs variable weighting to diversify the search and guide local search out of local minima.

However, each single local search heuristic is usually ineffective on many types of instances, since the performance of a heuristic depends on the properties of the search space and the search spaces of different types of instances have different properties. One way to design a local search algorithm that is effective on many types of instances is allowing this algorithm to switch among heuristics in order to adapt to search spaces with different properties.

Several local search algorithms switch between two heuristics [2,16]. $UnitWalk$ 0.98 [2] is improved by alternating between $WalkSAT$-like and $UnitWalk$-like fragments of the random walk. $Hybrid$ [16] switches between *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW according to the evenness or non-evenness of the distribution of variable weights.[1]

Nevertheless, our experimental results show that the performance of $Hybrid$ is poor on some instances for which a local search algorithm may result in imbalanced clause weights, and that the performance of clause weighting algorithm $RSAPS$ [6] is poor on some instances for which a local search algorithm may result in balanced clause weights. In this paper, we propose a new switching criterion: the evenness or unevenness of the distribution of clause weights. We refer to the ways in which non-weighting algorithm $adaptG^2WSAT+$ [15], clause weighting algorithm $RSAPS$, and variable weighting algorithm VW [13] select a variable to flip, as *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW, respectively. We apply this switching criterion together with another switching criterion, namely the evenness or non-evenness of the distribution of variable weights proposed in [16], to *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW. The resulting local search algorithm, which adaptively switches among these three heuristics in every search step according to these two criteria to intensify or diversify the search when necessary, is called $NCVW$ (Non-, Clause, and Variable Weighting).

Given a set of instances and a fixed cutoff for each instance, if an algorithm achieves a success rate greater than 50% for each instance, we say that this algorithm is generally effective on these instances. Our experimental results show that $NCVW$ is

[1] The ways in which algorithms $adaptG^2WSAT_P$ [9] and VW [13] select a variable to flip, are referred to as *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW, respectively [16].

generally effective on a wide range of instances while $adaptG^2WSAT+$, $RSAPS$, VW, $Hybrid$, and $gNovelty+$ and $adaptG^2WSAT0$, which won the gold and silver medals in the satisfiable random category in the SAT 2007 competition,[2] respectively, are not.

Our work for $NCVW$ provides a solution to the algorithm heuristic selection problem. Different approaches have been proposed for this problem. CBR (Case-Based Reasoning) was used to select a solution strategy for instances of a CP problem [1]. $SATzilla$-07 [17] is a per-instance solver portfolio for SAT. This solver portfolio uses machine learning techniques to build an empirical hardness model that predicts an algorithm's runtime on a given instance based on the features of this instance and the past performance of this algorithm, and uses this model to choose among the constituent solvers of $SATzilla$-07. $NCVW$ is different from $SATzilla$-07 in that $NCVW$ chooses heuristics for an instance dynamically during the search while $SATzilla$-07 first chooses an algorithm for an instance and then runs this algorithm on this instance.

2 Review of Algorithms $adaptG^2WSAT+$, $RSAPS$, VW, and $Hybrid$

The local search algorithm $adaptG^2WSAT+$ [15] combines the use of promising decreasing variables [7] and the adaptive noise mechanism [4]. As a result, noise p in this algorithm is adjusted during the search. Moreover, random walk probability wp is also adjusted and $wp = p/10$. This algorithm won the bronze medal in the satisfiable random category in the SAT 2007 competition. As presented in Section 1, we refer to the way in which algorithm $adaptG^2WSAT+$ selects a variable to flip, as *heuristic* $adaptG^2WSAT+$.

$SAPS$ [6] scales the weights of unsatisfied clauses and smoothes the weights of all clauses probabilistically. It performs a greedy descent search in which a variable is selected at random to flip, from the variables that appear in unsatisfied clauses and that lead to the maximum reduction in the total weight of unsatisfied clauses when flipped. $RSAPS$ [6] is a reactive version of $SAPS$ that adaptively tunes smoothing parameter P_{smooth} during the search. $RSAPS$ has the other parameters α, ρ, and wp whose default values are $(\alpha, \rho, wp) = (1.3, 0.8, 0.01)$. As presented in Section 1, we refer to the way in which algorithm $RSAPS$ selects a variable to flip, as *heuristic* $RSAPS$.

In VW [13], the weight of a variable reflects both the number of flips of this variable and the times when this variable is flipped. This algorithm initializes the weight of a variable x, $vw[x]$, to 0 and updates and smoothes $vw[x]$ each time x is flipped, using the following formula:

$$vw[x] = (1 - s)(vw[x] + 1) + s \times t \tag{1}$$

where s is a parameter and $0 \leq s \leq 1$, and t denotes the time when x is flipped, i.e., t is the number of search steps since the start of the search.

VW always flips a variable from a randomly selected unsatisfied clause c. If c contains freebie variables,[3] VW randomly flips one of them. Otherwise, with probability

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

[3] Flipping a freebie variable will not falsify any clause.

p (noise p), it flips a variable chosen randomly from c, and with probability $1 - p$, it flips a variable in c according to a unique variable selection rule, which generally favors variables with relatively low variable weights. As presented in Section 1, we refer to the way in which algorithm VW selects a variable to flip, as *heuristic* VW.

A switching criterion, namely the evenness or non-evenness of the distribution of variable weights, was proposed in [16]. This evenness or non-evenness is defined in [16] as follows. Assume that γ is a number. If the maximum variable weight is at least γ times as high as the average variable weight, the distribution of variable weights is considered *uneven*, and the step is called *an uneven step* in terms of variable weights. Otherwise, the distribution of variable weights is considered *even*, and the step is called *an even step* in terms of variable weights. An uneven or an even distribution of variable weights is used as a means to determine whether a search is undiversified in a step in terms of variable weights.

$Hybrid$ [16] switches between *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW according to the above switching criterion. More precisely, in each search step, $Hybrid$ chooses a variable to flip according to *heuristic* VW if the distribution of variable weights is uneven, and selects a variable to flip according to *heuristic* $adaptG^2WSAT_P$ otherwise. In $Hybrid$, the default value of parameter γ is 10.0. $Hybrid$ updates variable weights using Formula 1, and parameter s in this formula is fixed to 0.0.

3 Motivation

The search space of a hard SAT instance for a local search algorithm generally has a large number of local minima in each of which flipping any variable cannot decrease the number of unsatisfied clauses. Each local minimum is characterized by a subset of unsatisfied clauses, which cannot be satisfied without unsatisfying other clauses. The unsatisfied clauses in a local minimum can be considered as having attractions to draw a local search towards this local minimum. Different clauses in a SAT instance can have very different attractions for a local search. A local search can be frequently drawn towards the same local minima by the same unsatisfied clauses with strong attractions. In this case, the search is poorly diversified. Accordingly, clause weighting techniques are introduced to diversify this poorly diversified search.

Clause weighting in a local search algorithm has two purposes. The first one is to quantify the attraction of a clause for a local search. Different clause weights are defined to measure the attractions of clauses for local searches [6,11]. The second one is to modify the objective function, which is usually the number of unsatisfied clauses, during the search by minimizing the total weight of unsatisfied clauses instead of the number of unsatisfied clauses. As a result, a clause weighting algorithm usually first satisfies clauses that have the largest attractions to diversify the search.

We hypothesize firstly that, without clause weighting techniques, $Hybrid$ [16] exhibits good performance, usually when clause weights are generally balanced, i.e., usually when all clauses have roughly equal attractions for a local search towards local minima. We hypothesize secondly that, with clause weighting techniques, $RSAPS$ [6] exhibits good performance, usually when clause weights are unbalanced. To empirically verify our hypotheses, we conduct experiments with $Hybrid$ and $RSAPS$ on

two classes of representative instance, one of which leads to balanced clauses weights and the other of which leads to unbalanced clause weights.

In order to quantify the attraction of a clause towards a local minimum at a search point, we simply sum up the number of local minima, encountered so far, in which this clause is unsatisfied. In fact, these summations of the numbers of local minima are the clause weights defined in $Breakout$ [11]. We refer to these clause weights as *clause weights defined by* $Breakout$. We calculate these clause weights in both $Hybrid$ and $RSAPS$ to make clause weights comparable for these two algorithms, although $RSAPS$ has its own clause weighting techniques, which are more sophisticated. Note that the calculations of these clause weights in these two algorithms do not change the behavior of $Hybrid$ or $RSAPS$ in any way, i.e., these two algorithms do not consider the clause weights that we calculate, when choosing variables to flip.

We ran $RSAPS$ and $Hybrid$ on two classes of instance.[4] The source code of $RSAPS$ was downloaded from http://www.satlib.org/ubcsat/. When experimenting with these algorithms, we do not change the ways in which these algorithms adaptively adjust their parameters and do not change the default values of their other parameters. One class includes the 5 structured instances par16-1, par16-2, par16-3, par16-4, and par16-5 in PARITY from the SATLIB repository,[5] and the other consists of the 5 structured instances f*3995, f*3997, f*3999, f*4001, and f*4003 in Ferry from the industrial category in the SAT 2005 competition benchmark.[6] Each algorithm is run 100 times ($Maxtries = 100$). The cutoffs are set to 10^9 ($Maxsteps = 10^9$) and 10^8 ($Maxsteps = 10^8$) for an instance in PARITY and an instance in Ferry, respectively.

In Table 1, for each of the two algorithms $Hybrid$ and $RSAPS$, we report the average coefficient of variation of distribution of clause weights (coefficient of variation = standard deviation / mean value) ("cv") and the average division of the maximum clause weight by the average clause weight ("div"), over all search steps, for clause weights defined by $Breakout$. All reported values are then averaged over 100 runs ($Maxtries = 100$). A run is successful if an algorithm finds a solution within a cutoff ($Maxsteps$). The success rate of an algorithm is the number of successful runs divided by $Maxtries$. We also report success rates ("suc"). Moreover, in the last row for each group ("avg") in this table, we present the average of the values in each column, for each algorithm, over all instances.

Generally speaking, the distribution of clause weights defined by $Breakout$ reflects the history of the attractions of clauses for local searches towards local minima. If clauses in a small subset have drawn local searches towards local minima much more frequently than other clauses, the weights of these clauses should be much higher than those of others and the coefficient of variation of distribution of clause weights should be high. Otherwise, all clauses should have approximately equal weights, and the coefficient of variation of distribution of clause weights should be low. That is, the higher the coefficient of variation is, the more clause weights far from the mean value exist,

[4] All experiments reported were conducted on a computer with Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz and with 2GB of memory, under Linux.

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

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

Table 1. Performance and distributions of clause weights defined by $Breakout$ for $RSAPS$ and $Hybrid$ on PARITY and Ferry

		$RSAPS$			$Hybrid$		
	cutoff	cv	div	suc	cv	div	suc
par16-1	10^9	0.82	7.22	0.19	10.20	127.06	1.00
par16-2	10^9	0.82	7.31	0.05	10.40	131.58	1.00
par16-3	10^9	0.82	7.27	0.12	10.30	129.04	1.00
par16-4	10^9	0.82	7.22	0.16	10.23	127.75	1.00
par16-5	10^9	0.82	7.28	0.09	10.43	131.88	1.00
avg	10^9	0.82	7.26	0.12	10.31	126.46	1.00
f*3995	10^8	2.72	22.38	1.00	41.05	1910.94	0.06
f*3997	10^8	2.06	12.15	1.00	47.51	2632.37	0.02
f*3999	10^8	2.36	17.39	1.00	48.23	2594.44	0.00
f*4001	10^8	2.13	12.80	0.84	55.04	3489.75	0.00
f*4003	10^8	2.11	13.57	1.00	55.35	3375.36	0.00
avg	10^8	2.28	15.66	0.97	49.44	2800.57	0.02

the more unbalanced cause weights are, and the less well diversified, in terms of clause weights, the search is.

According to Table 1, the distributions of clause weights of PARITY are more balanced than the distributions of clause weights of Ferry, both for $Hybrid$ and for $RSAPS$, meaning that when solving the Ferry instances, some clauses in the Ferry instances frequently draw local searches towards local minima so that the searches for Ferry are less diversified than those for PARITY. Nevertheless, with its own powerful clause weighting techniques, $RSAPS$ diversifies the search better than does $Hybrid$. As a result, the distributions of clause weights in the searches of $RSAPS$ are more balanced for each of the two classes of instance than the distributions of clause weights in the searches of $Hybrid$.

As shown in Table 1, the average success rate of $Hybrid$ on PARITY is 1.00, suggesting that when the distributions of clause weights are generally balanced, $Hybrid$, which does not use clause weighting techniques, exhibits good performance. However, the average success rate of $Hybrid$ on Ferry is only 0.02, suggesting that when the distributions of clause weights are generally unbalanced, the performance of $Hybrid$ is poor. Conversely, $RSAPS$ exhibits very good performance on Ferry but poor performance on PARITY. Therefore, the experimental results in this table indicate that an algorithm should ignore clause weights and concentrate on intensifying the search if clause weights are generally balanced, and that an algorithm should use clause weighting techniques, such as those introduced in $RSAPS$, to diversify the search to prevent a local search from being drawn towards the same local minima.

According to Table 1, on PARITY, the averages of the values for div in $RSAPS$ and $Hybrid$ are 7.26 and 126.46, respectively, while on Ferry, the averages of the values for div in $RSAPS$ and $Hybrid$ are 15.66 and 2800.57, respectively. That is, the maximum clause weight on Ferry usually deviates from the average clause weight to a greater degree than does the maximum clause weight on PARITY. Therefore, the results in this table suggest that, similar to the coefficient of variation of distribution of clause weights, the division of the maximum clause weight by the average clause weight also indicates whether clause weights are balanced. In fact, calculating the division is not time-consuming, but calculating the coefficient of variation is.

4 A New Switching Criterion

We propose a new switching criterion. Additionally, we introduce a new local search algorithm that uses this criterion along with another switching criterion.

4.1 Evenness or Unevenness of Distribution of Clause Weights

Assume that δ is a number. If the maximum clause weight is at least δ times as high as the average clause weight, the distribution of clause weights is considered *uneven*, and the step is called *an uneven step* in terms of clause weights. Otherwise, the distribution of clause weights is considered *even*, and the step is called *an even step* in terms of clause weights. An uneven distribution and an even distribution of clause weights correspond to the situations in which clause weights are unbalanced and balanced, respectively. We use an uneven or an even distribution of clause weights as a means to determine whether a search is undiversified in a step in terms of clause weights.

4.2 Algorithm $NCVW$

To evaluate the effectiveness of the proposed switching criterion, we apply it together with another switching criterion, namely the evenness or non-evenness of the distribution of variable weights proposed in [16], to *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW. The resulting local search algorithm, which switches among these three heuristics according to these two criteria, is called $NCVW$ (Non-, Clause, and Variable Weighting).

$NCVW$ exploits the information about the structure of an instance when choosing a variable to flip by first examining the distribution of variable weights of this instance and then examining the distribution of clause weights of this instance. This algorithm adaptively switches among *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW in every search step according to the distributions of variable and clause weights, to intensify or diversify the search when necessary. When the distribution of variable weights is uneven, i.e., when a search is undiversified in terms of variable weights, $NCVW$ uses *heuristic* VW to choose a variable to flip to diversify the search by using variable weights. Otherwise, $NCVW$ selects a variable to flip according to *heuristic* $RSAPS$ or *heuristic* $adaptG^2WSAT+$, depending on whether the distribution of clause weights is uneven. If the distribution of clause weights is uneven, i.e., if a search is undiversified in terms of clause weights, $NCVW$ uses *heuristic* $RSAPS$ to select a variable to flip to diversify the search by using clause weights; otherwise, i.e., if a search is diversified in terms of both variable and clause weights, $NCVW$ uses *heuristic* $adaptG^2WSAT+$ to select a variable to flip to intensify the search.

$NCVW$ is described in Fig. 1. In this figure, $flip\_time[i]$, $vw[i]$, max_vw, ave_vw, $cw[j]$, max_cw, and ave_cw are the time when variable i is flipped, the weight of variable i, maximum variable weight, average variable weight, the weight of clause j, maximum clause weight, and average clause weight, respectively.

$NCVW$ has its own parameters γ, δ, and π, which are used to choose one heuristic from $NCVW$'s constituent heuristics in every step. Parameter γ determines whether

Algorithm: $NCVW$(SAT-formula $\mathcal{F}$)

1: $A \leftarrow$ randomly generated truth assignment;
2: **for** each variable i **do** initialize $flip\_time[i]$ and $vw[i]$ to 0;
3: initialize max_vw and ave_vw to 0;
4: **for** each clause j **do** initialize $cw[j]$ to 1; initialize max_cw and ave_cw to 1;
5: **for** *flip*$\leftarrow$1 **to** *Maxsteps* **do**
6: **if** A satisfies $\mathcal{F}$ **then return** A;
7: **if** $(max\_vw \geq \gamma \times ave\_vw)$
8: **then** $heuristic \leftarrow$ "VW";
9: **else**
10: **if** $((ave\_cw \leq \pi)$ or $(max\_cw \geq \delta \times ave\_cw))$
11: **then** $heuristic \leftarrow$ "$RSAPS$";
12: **else** $heuristic \leftarrow$ "$adaptG^2WSAT+$";
13: $y \leftarrow$ use *heuristic* to choose a variable;
14: **if** $(y \neq -1)$
15: **then** $A \leftarrow A$ with y flipped; update $flip\_time[y]$, $vw[y]$, max_vw, and ave_vw;
16: **if** $(heuristic =$ "$RSAPS$")
17: **then if** $(y = -1)$ **then** update clause weights, max_cw, and ave_cw;
18: **return** Solution not found;

Fig. 1. Algorithm $NCVW$

the distribution of variable weights is uneven, δ determines whether the distribution of clause weights is uneven, and π represents a threshold for average clause weight.

In algorithm $NCVW$, *heuristic* $RSAPS$ is used both for gathering information about the distribution of clause weights and for selecting a variable to flip when the distribution of clause weights is uneven. In this algorithm, the n variables of an instance are represented as n integers from 0 to $n-1$. When $NCVW$ uses *heuristic* $RSAPS$ and when this heuristic returns -1, $NCVW$ performs a null flip and updates clause weights in the same way as does $RSAPS$. To avoid frequent time-consuming clause weight updating, $NCVW$ does not update clause weights if it uses *heuristic* $adaptG^2WSAT+$ or *heuristic* VW. $NCVW$ uses Formula 1 to update variable weights after it selects any heuristic from its three constituent heuristics and after $NCVW$ performs a non-null flip.

We have three objectives in $NCVW$. The first objective is to ensure that every variable in a SAT instance has an approximately equal chance of being flipped to diversify the search in terms of variable weights, i.e., to ensure that the distribution of variable weights is even. When the distribution of variable weights is uneven, $NCVW$ chooses heuristic VW to balance variable weights. Whether the distribution of variable weights is even is determined at line 7 using the condition $(max\_vw \geq \gamma \times ave\_vw)$, in which parameter $\gamma > 1.0$, in Fig. 1.

The second objective is to avoid the same local minima or to avoid exploring the same regions in the search space, i.e., to ensure that the distribution of clause weights is even. Since $NCVW$ updates clause weights only when *heuristic* $RSAPS$ is used to choose a variable to flip, parameter π, which represents a threshold for average clause weight ave_cw, is introduced. When the distribution of variable weights is even,

$NCVW$ chooses *heuristic* $RSAPS$ to build up the distribution of clause weights whenever the average clause weight is lower than parameter π. When the average clause weight is higher than π, clause weights are considered meaningful to determine whether the distribution of clause weights is even, and *heuristic* $RSAPS$ is chosen to balance the distribution of clause weights and to diversify the search in terms of clause weights if the distribution of clause weights is uneven. This building up of distribution of clause weights and this balancing of distribution of clause weights are realized through line 10 using the condition $((ave\_cw \leq \pi)$ or $(max\_cw \geq \delta \times ave\_cw))$, in which parameter $\delta > 1.0$, in Fig. 1.

The third objective is to intensify the search when the distributions of both variable and clause weights are even. In this case, the search is considered diversified in terms of both variable and clause weights, and *heuristic* $adaptG^2WSAT+$ is used to intensify the search.

$NCVW$ is an example that uses the proposed switching criterion along with the switching criterion proposed in [16]. These switching criteria can be used in other local search algorithms that combine intensification strategies with diversification strategies.

5 Evaluation

We present the default values of parameters in $NCVW$. Moreover, we evaluate $NCVW$ on a wide range of instances and justify the switching strategy in $NCVW$.

5.1 Groups of Instances

We evaluate $NCVW$ on 11 groups of benchmark SAT problems (36 instances shown in Table 2). They generally consist of hard problems from those widely used to evaluate local search algorithms in the literature and constitute a wide range of instances, including structured instances, instances from the industrial and crafted categories in a SAT competition benchmark, and hard random instances. Due to space limits, we do not present the performance of algorithms on the instances that are easy for most algorithms discussed in this paper. Structured problems come from the SATLIB repository and the SAT 2005 competition benchmark. The structured problems from SATLIB include instances in ais, blocksworld, Beijing, GCP, PARITY, and QG. The structured problems from the SAT 2005 competition benchmark include instances from the industrial category and instances from the crafted category. The former consist of f*3995, f*3997, f*3999, f*4001, and f*4003 in Ferry. The latter consist of g*1334, g*1337, g*1339, g*1340, and g*1341 in grid-pebbling/sat, and p*1318, p*1319, p*1320, p*1321, and p*1322 in random-pebbling/sat. Random problems come from the SAT 2007 competition benchmark,[7] and they are hard problems, including *v10000*03, *v10000*04, *v10000*05, *v10000*06, and *v10000*10 in 3SAT/v10000, and *v1100*04, *v1100*06, *v1100*08, *v1100*10, and *v1100*14 in 5SAT/v1100.

Each instance is executed 100 times ($Maxtries = 100$). As shown in Table 2, the search step cutoff ($Maxsteps$) for each instance is set to a fixed value, to ensure that

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

Table 2. Experimental results for $NCVW$, $adaptG^2WSAT+$, $RSAPS$, and VW on the 11 groups of instances

		$NCVW$			$adaptG^2WSAT+$			$RSAPS$			VW		
	cutoff	suc	#steps	time	suc	#steps	time	suc	#steps	time	suc	#steps	time
ais12	10^7	0.93	249576	**0.2**	0.94	2568069	2.4	**1.00**	**159617**	**0.2**	**1.00**	962263	1.4
bw_large.d	10^7	**0.96**	**955185**	**2.5**	0.70	5786347	8.1	0.06	$> 10^7$	n/a	0.94	2554959	5.4
e0ddr2*1	10^7	**1.00**	**119631**	**0.5**	0.96	2219658	3.3	**1.00**	120139	0.7	0.71	6656635	10.7
g250.29	10^7	0.79	3483743	76.9	**1.00**	**728358**	**6.3**	0.00	$> 10^7$	n/a	0.15	$> 10^7$	n/a
par16-1	10^9	**1.00**	96371225	48.3	**1.00**	**45395657**	**14.9**	0.16	$> 10^9$	n/a	0.00	$> 10^9$	n/a
par16-2	10^9	0.88	329217086	166.7	**1.00**	**96745059**	**32.5**	0.07	$> 10^9$	n/a	0.00	$> 10^9$	n/a
par16-3	10^9	**1.00**	115526876	58.8	**1.00**	**87203523**	**28.9**	0.08	$> 10^9$	n/a	0.00	$> 10^9$	n/a
par16-4	10^9	0.96	183854606	92.8	**0.98**	**170022013**	**55.7**	0.10	$> 10^9$	n/a	0.01	$> 10^9$	n/a
par16-5	10^9	0.93	264988982	133.5	**0.99**	**111417125**	**37.0**	0.13	$> 10^9$	n/a	0.00	$> 10^9$	n/a
qg2-08	10^7	0.83	2832886	12.7	**1.00**	**1766643**	**4.9**	0.08	$> 10^7$	n/a	0.12	$> 10^7$	n/a
qg7-13	10^8	**1.00**	**3663127**	**31.9**	0.20	$> 10^8$	n/a	0.07	$> 10^8$	n/a	0.00	$> 10^8$	n/a
f*3995	10^8	**1.00**	**56405**	**0.1**	0.04	$> 10^8$	n/a	**1.00**	66907	**0.1**	**1.00**	5346194	4.2
f*3997	10^8	0.95	**4830154**	**3.3**	0.01	$> 10^8$	n/a	**1.00**	6348614	5.8	0.69	55768455	29.8
f*3999	10^8	**1.00**	336040	0.4	0.00	$> 10^8$	n/a	**1.00**	**269670**	**0.3**	0.38	$> 10^8$	n/a
f*4001	10^8	0.66	51971463	41.8	0.00	$> 10^8$	n/a	**0.80**	**39945421**	**40.3**	0.18	$> 10^8$	n/a
f*4003	10^8	**1.00**	**1514236**	**1.9**	0.00	$> 10^8$	n/a	**1.00**	1575665	**1.9**	0.05	$> 10^8$	n/a
g*1334	10^8	**1.00**	248515	0.2	0.11	$> 10^8$	n/a	**1.00**	385746	0.2	**1.00**	**165058**	**0.1**
g*1337	10^8	**1.00**	1411026	1.2	0.51	52247390	24.7	**1.00**	4188026	3.1	**1.00**	**334304**	**0.2**
g*1339	10^8	**1.00**	2325027	3.5	0.70	1656358	1.9	**1.00**	19332961	27.5	**1.00**	**1090780**	**1.1**
g*1340	10^8	0.76	**3579295**	**7.3**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	**1.00**	9592247	10.3
g*1341	10^8	0.98	**4507419**	10.3	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	**1.00**	6911453	**8.5**
p*1318	10^8	**1.00**	1411992	6.2	0.19	$> 10^8$	n/a	**1.00**	1109307	**5.3**	**1.00**	**1023732**	37.8
p*1319	10^8	**1.00**	1039612	4.1	0.64	1283571	3.5	**1.00**	361676	**0.9**	**1.00**	**202640**	2.3
p*1320	10^8	**1.00**	4264494	13.4	0.00	$> 10^8$	n/a	**1.00**	3167791	**8.7**	**1.00**	**555636**	9.1
p*1321	10^8	**1.00**	8401861	27.4	0.01	$> 10^8$	n/a	**1.00**	3512466	**6.8**	**1.00**	**925001**	12.8
p*1322	10^8	0.94	18686150	85.0	0.02	$> 10^8$	n/a	0.99	11563065	**39.5**	**1.00**	**1203442**	45.8
*v10000*03	10^9	0.93	195343618	310.3	0.38	$> 10^9$	n/a	0.00	$> 10^9$	n/a	**1.00**	**51325928**	**70.1**
*v10000*04	10^9	0.78	445825231	703.0	0.12	$> 10^9$	n/a	0.00	$> 10^9$	n/a	**1.00**	**73729009**	**99.9**
*v10000*05	10^9	0.56	928751054	1407.3	0.00	$> 10^9$	n/a	0.00	$> 10^9$	n/a	**0.96**	**140754923**	**188.4**
*v10000*06	10^9	0.92	287233505	427.6	0.22	$> 10^9$	n/a	0.00	$> 10^9$	n/a	**0.99**	**52498950**	**70.6**
*v10000*10	10^9	0.84	340420147	536.8	0.10	$> 10^9$	n/a	0.00	$> 10^9$	n/a	**0.99**	**71314104**	**95.9**
*v1100*04	10^9	**0.99**	160716875	692.8	**0.99**	**154199134**	**454.7**	0.00	$> 10^9$	n/a	0.00	$> 10^9$	n/a
*v1100*06	10^9	0.96	254443055	1082.3	**0.97**	**181909223**	**538.3**	0.00	$> 10^9$	n/a	0.00	$> 10^9$	n/a
*v1100*08	10^9	**0.94**	**271460435**	1159.5	**0.94**	274281212	**810.0**	0.00	$> 10^9$	n/a	0.00	$> 10^9$	n/a
*v1100*10	10^9	**0.98**	**201335004**	852.7	0.94	244056999	**723.9**	0.00	$> 10^9$	n/a	0.00	$> 10^9$	n/a
*v1100*14	10^9	**1.00**	**48218593**	205.9	**1.00**	56308880	**166.8**	0.00	$> 10^9$	n/a	0.02	$> 10^9$	n/a

at least one algorithm discussed achieves a success rate greater than 50% in order to calculate median number of search steps and median run time based on these 100 runs. We report success rates ("suc"), median numbers of search steps ("#steps"), and median run times ("time") in seconds. If an algorithm cannot achieve a success rate greater than 50% on an instance within the specified cutoff, we use "$> Maxsteps$" (greater than $Maxsteps$) and "n/a" to denote the median number of search steps and median run time, respectively. Results in bold indicate the best results for an instance.

5.2 Default Values of Parameters in $NCVW$

Like VW, $NCVW$ updates variable weights using Formula 1. To adapt to $NCVW$, which is set up to solve a wide range of instances, parameter s in this formula is fixed to 0.0. When s is 0.0, the weight of a variable is just a counter of the number of flips of this variable. Conversely, in VW, s is adjusted during the search ($s > 0.0$). That

is, $NCVW$ does not smooth variable weights while VW does. This non-smoothing of variable weights makes uneven distributions of variable weights and even distributions of variable weights more distinguishable.

In addition to its own parameters γ, δ, and π, $NCVW$ has all parameters from its constituent heuristics. According to our experiments, on a wide range of instances, $NCVW$ with $(\gamma, \delta, \pi, s, wp) = (7.5, 3.0, 15.0, 0.0, 0.05)$ (wp is from $RSAPS$) exhibits generally good performance. Thus, in $NCVW$, the default values of γ, δ, π, s, and wp are $(\gamma, \delta, \pi, s, wp) = (7.5, 3.0, 15.0, 0.0, 0.05)$. For the other parameters in $NCVW$ from its constituent heuristics, $NCVW$ adaptively adjusts these parameters as do the constituent heuristics or uses the same default values as do the constituent heuristics.

5.3 Comparison of Performance

We compare the performance of $NCVW$, $adaptG^2WSAT+$, $RSAPS$, and VW on the 11 groups of instances (36 instances) in Table 2, and compare the performance of $NCVW$, $gNovelty+$, $adaptG^2WSAT0$, and $Hybrid$ on these instances in Table 3. The source code of $adaptG^2WSAT+$, $gNovelty+$, and $adaptG^2WSAT0$ was downloaded from http://www.satcompetition.org/, and that of $RSAPS$ was downloaded from http://www.satlib.org/ubcsat/. The source code of VW was obtained from the organizer of the SAT 2005 competition. When experimenting with these algorithms, we do not change the ways in which these algorithms adaptively adjust their parameters and do not change the default values of the other parameters in these algorithms either.

According to our experiments, the 36 instances include those that, for $NCVW$, usually lead to the following four combinations of the distributions of variable and clause weights: the distributions of both variable and clause weights are even, the distributions of variable weights are even while the distributions of clause weights are uneven, the distributions of variable weights are uneven while the distributions of clause weights are even, and the distributions of both variable and clause weights are uneven. Specifically, for $NCVW$, the instances in PARITY and 5SAT/v1100 generally result in even distributions of both variable and clause weights. The instances in Ferry and QG usually lead to even distributions of variable weights but uneven distributions of clause weights. The instances g*1340 and g*1341 in grid-pebbling/sat usually result in uneven distributions of variable weights but even distributions of clause weights. The instances in blocksworld and Beijing generally lead to uneven distributions of both variable and clause weights.

According to Table 2, within the specified cutoffs, $NCVW$ is generally effective on these 11 groups. Conversely, within the designated cutoffs, $adaptG^2WSAT+$, $RSAPS$, and VW are effective on only 6, 4, and 6 groups, respectively.

$NCVW$ exhibits good performance on qg7-13 although $adaptG^2WSAT+$, $RSAPS$, VW all show poor performance on this instance. There are two reasons for this good performance. First, like $adaptG^2WSAT+$, $NCVW$ conducts preprocessing using unit propagation to simplify an instance before searching. Second, $NCVW$ usually chooses *heuristic* $RSAPS$ automatically to select a variable to flip for the simplified qg7-13 because this simplified qg7-13 results in even distribution of variable

Table 3. Experimental results for $NCVW$, $gNovelty+$, $adaptG^2WSAT0$, and $Hybrid$ on the 11 groups of instances

		$NCVW$			$gNovelty+$			$adaptG^2WSAT0$			$Hybrid$		
	cutoff	suc	#steps	time	suc	#steps	time	suc	#steps	time	suc	#steps	time
ais12	10^7	0.93	**249576**	**0.2**	0.32	$> 10^7$	n/a	**1.00**	1181980	1.1	**1.00**	1534609	2.0
bw_large.d	10^7	**0.96**	955185	2.5	0.60	6561933	16.7	0.49	$> 10^7$	n/a	**0.96**	**661253**	**2.1**
e0ddr2*1	10^7	**1.00**	119631	**0.5**	0.00	$> 10^7$	n/a	0.96	2013651	3.0	**1.00**	**117320**	1.7
g250.29	10^7	0.79	3483743	76.9	**1.00**	**590941**	10.7	**1.00**	806380	**7.0**	0.88	1360491	22.6
par16-1	10^9	**1.00**	96371225	48.3	0.51	985564819	231.1	**1.00**	78289718	**26.1**	**1.00**	**69932292**	29.6
par16-2	10^9	0.88	329217086	166.7	0.38	$> 10^9$	n/a	**0.99**	**105017111**	**35.8**	**0.99**	119600333	52.2
par16-3	10^9	**1.00**	115526876	58.8	0.52	955707228	224.5	**1.00**	113814551	**39.0**	**1.00**	**92166515**	40.2
par16-4	10^9	0.96	183854606	92.8	0.26	$> 10^9$	n/a	0.98	142059581	47.7	**0.99**	**95694408**	**40.8**
par16-5	10^9	0.93	264988982	133.5	0.55	924073354	219.7	**1.00**	113484583	38.8	0.99	**81990858**	**35.4**
qg2-08	10^7	0.83	2832886	12.7	0.02	$> 10^7$	n/a	0.98	1757920	**4.8**	**0.99**	**1440339**	6.1
qg7-13	10^8	**1.00**	**3663127**	**31.9**	0.00	$> 10^8$	n/a	0.20	$> 10^8$	n/a	0.40	$> 10^8$	n/a
f*3995	10^8	**1.00**	**56405**	**0.1**	0.00	$> 10^8$	n/a	0.06	$> 10^8$	n/a	0.15	$> 10^8$	n/a
f*3997	10^8	**0.95**	**4830154**	**3.3**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.01	$> 10^8$	n/a
f*3999	10^8	**1.00**	**336040**	**0.4**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a
f*4001	10^8	**0.66**	**51971463**	**41.8**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a
f*4003	10^8	**1.00**	**1514236**	**1.9**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a
g*1334	10^8	**1.00**	248515	0.2	0.04	$> 10^8$	n/a	0.01	$> 10^8$	n/a	**1.00**	**69867**	**0.1**
g*1337	10^8	**1.00**	1411026	1.2	0.47	$> 10^8$	n/a	0.31	$> 10^8$	n/a	**1.00**	**146414**	**0.2**
g*1339	10^8	**1.00**	2325027	3.5	0.67	23436378	32.7	0.49	$> 10^8$	n/a	**1.00**	**602498**	**2.0**
g*1340	10^8	0.76	3579295	7.3	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	**1.00**	**2542104**	**6.4**
g*1341	10^8	0.98	4507419	10.3	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a	**1.00**	**2875184**	**9.8**
p*1318	10^8	**1.00**	1411992	6.2	0.49	$> 10^8$	n/a	0.05	$> 10^8$	n/a	0.85	**828436**	**4.2**
p*1319	10^8	**1.00**	**1039612**	**4.1**	0.16	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.26	$> 10^8$	n/a
p*1320	10^8	**1.00**	4264494	13.4	0.63	58348721	135.6	0.03	$> 10^8$	n/a	0.79	**507350**	**2.2**
p*1321	10^8	**1.00**	**8401861**	**27.4**	0.02	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.15	$> 10^8$	n/a
p*1322	10^8	**0.94**	**18686150**	**85.0**	0.09	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.17	$> 10^8$	n/a
*v10000*03	10^9	0.93	195343618	310.3	**1.00**	**55787237**	**93.0**	0.12	$> 10^9$	n/a	0.98	162732471	229.1
*v10000*04	10^9	0.78	445825231	703.0	**1.00**	**69007926**	**111.5**	0.02	$> 10^9$	n/a	0.88	262272862	345.4
*v10000*05	10^9	0.56	928751054	1407.3	**1.00**	**134839615**	**219.3**	0.00	$> 10^9$	n/a	0.77	516289640	735.8
*v10000*06	10^9	0.92	287233505	427.6	**1.00**	**53678135**	**84.1**	0.10	$> 10^9$	n/a	**1.00**	134103486	208.1
*v10000*10	10^9	0.84	340420147	536.8	**1.00**	**69535298**	**110.0**	0.00	$> 10^9$	n/a	0.92	260197067	370.4
*v1100*04	10^9	0.99	160716875	692.8	0.04	$> 10^9$	n/a	**1.00**	**134579805**	**390.1**	0.88	298792644	1410.6
*v1100*06	10^9	0.96	254443055	1082.3	0.08	$> 10^9$	n/a	**0.98**	**189797513**	**544.4**	0.75	468379966	2194.6
*v1100*08	10^9	**0.94**	271460435	1159.5	0.08	$> 10^9$	n/a	**0.94**	**219521877**	**627.1**	0.69	643453270	3039.1
*v1100*10	10^9	**0.98**	**201335004**	852.7	0.06	$> 10^9$	n/a	0.96	219480345	**629.1**	0.73	547026001	2556.3
*v1100*14	10^9	**1.00**	**48218593**	205.9	0.26	$> 10^9$	n/a	**1.00**	54801707	**157.2**	**1.00**	112659869	525.2

weights and uneven distribution of clause weights, and *heuristic* $RSAPS$ is effective on the simplified qg7-13 although this heuristic is not effective on the original qg7-13.

As shown in Table 3, within the specified cutoffs, $NCVW$ is generally effective on these 11 groups of instances while $gNovelty+$, $adaptG^2WSAT0$, and $Hybrid$ are effective on only 3, 5, and 8 groups, respectively.

5.4 Justification for Switching Strategy Used in $NCVW$

To justify the proposed switching strategy used in $NCVW$, we implement two other switching strategies in two algorithms $NCVW\_diff$ and $NCVW\_rand$, which are described as follows. If the distribution of variable weights is uneven or the distribution of clause weights is uneven, $NCVW\_diff$ chooses a variable to flip according to *heuristic* $adaptG^2WSAT+$. Otherwise, i.e., if the distributions of both variable and

Table 4. Experimental results for $NCVW$, $NCVW\_diff$, and $NCVW\_rand$ on the hardest instances in the 11 groups

		$NCVW$			$NCVW\_diff$			$NCVW\_rand$		
	cutoff	suc	#steps	time	suc	#steps	time	suc	#steps	time
ais12	10^7	0.93	**249576**	**0.2**	0.94	3068730	4.5	**1.00**	340597	0.5
bw_large.d	10^7	**0.96**	**955185**	**2.5**	0.70	7040081	16.2	0.90	2061289	5.2
e0ddr2*1	10^7	**1.00**	**119631**	**0.5**	0.96	2206478	4.3	**1.00**	193154	0.7
g250.29	10^7	0.79	3483743	76.9	**1.00**	**771854**	**11.6**	**1.00**	994184	13.9
par16-2	10^9	0.88	329217086	166.7	**1.00**	**131330419**	**65.8**	0.98	198219412	118.2
qg7-13	10^8	**1.00**	**3663127**	**31.9**	0.20	$> 10^8$	n/a	0.32	$> 10^8$	n/a
f*4001	10^8	**0.66**	**51971463**	**41.8**	0.00	$> 10^8$	n/a	0.00	$> 10^8$	n/a
g*1341	10^8	0.98	**4507419**	**10.3**	0.00	$> 10^8$	n/a	**1.00**	28817283	66.2
p*1322	10^8	0.94	18686150	85.0	0.01	$> 10^8$	n/a	**1.00**	**1932060**	**18.5**
*v10000*05	10^9	**0.56**	**928751054**	**1407.3**	0.02	$> 10^9$	n/a	0.14	$> 10^9$	n/a
*v1100*08	10^9	**0.94**	271460435	1159.5	**0.94**	**253006754**	**1056.5**	0.39	$> 10^9$	n/a

clause weights are even, $NCVW\_diff$ first randomly selects a heuristic from *heuristic* $RSAPS$ and *heuristic* VW, and then chooses a variable to flip according to the randomly selected heuristic. In each search step, $NCVW\_rand$ randomly selects a heuristic from *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW, and then uses this randomly selected heuristic to choose a variable to flip.

Both $NCVW\_diff$ and $NCVW\_rand$ update variable and clause weights in the same ways as does $NCVW$. For an instance that leads to even distributions of variable weights, $NCVW\_diff$ will build up the distribution of clause weights, after a small number of search steps compared with the total search steps that $NCVW\_diff$ performs for this instance. Thus, $NCVW\_diff$ does not need parameter π. For each of the other parameters in $NCVW\_diff$, $NCVW\_diff$ adjusts this parameter as does $NCVW$ or uses the same default value as does $NCVW$. As opposed to $NCVW$, $NCVW\_rand$ does not need parameters γ, δ, and π. For each of the other parameters in $NCVW\_rand$, $NCVW\_rand$ adjusts this parameter as does $NCVW$ or uses the same default value as does $NCVW$. In Table 4, we compare the performance of $NCVW$, $NCVW\_diff$, and $NCVW\_rand$ on the hardest instances in the 11 groups for most algorithms discussed in this paper. Within the specified cutoffs, $NCVW$ is generally effective on these 11 instances, but $NCVW\_diff$ and $NCVW\_rand$ are effective on only 6 and 7 instances, respectively.

6 Conclusion

We have proposed a new switching criterion: the evenness or unevenness of the distribution of clause weights. We apply this criterion, along with another switching criterion, to *heuristic* $adaptG^2WSAT+$, *heuristic* $RSAPS$, and *heuristic* VW. The resulting algorithm, which combines intensification strategies with diversification strategies, is called $NCVW$ (Non-, Clause, and Variable Weighting). Experimental results show that $NCVW$ is generally effective on a wide range of instances whereas $adaptG^2WSAT+$, $RSAPS$, VW, $gNovelty+$, $adaptG^2WSAT0$, and $Hybrid$ are not.

References

1. Gebruers, C., Hnich, B., Bridge, D.G., Freuder, E.C.: Using CBR to Select Solution Strategies in Constraint Programming. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 222–236. Springer, Heidelberg (2005)
2. Hirsch, E.A., Kojevnikov, A.: UnitWalk: A New SAT Solver that Uses Local Search Guided by Unit Clause Elimination. Ann. Math. Artif. Intell. 43(1), 91–111 (2005)
3. Hoos, H.H.: On the Run-Time Behavior of Stochastic Local Search Algorithms for SAT. In: Proceedings of AAAI 1999, pp. 661–666. AAAI Press, Menlo Park (1999)
4. Hoos, H.H.: An Adaptive Noise Mechanism for WalkSAT. In: Proceedings of AAAI 2002, pp. 655–660. AAAI Press, Menlo Park (2002)
5. Hoos, H.H., Stűtzle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann, San Francisco (2004)
6. Hutter, F., Tompkins, D.A.D., Hoos, H.H.: Scaling and Probabilistic Smoothing: Efficient Dynamical Local Search for SAT. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 233–248. Springer, Heidelberg (2002)
7. Li, C.M., Huang, W.Q.: Diversification and Determinism in Local Search for Satisfiability. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 158–172. Springer, Heidelberg (2005)
8. Li, C.M., Wei, W., Zhang, H.: Combining Adaptive Noise and Promising Decreasing Variables in Local Search for SAT, `http://www.satcompetition.org/2007/contestants.html`
9. Li, C.M., Wei, W., Zhang, H.: Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In: Benhamou, F., Jussien, N., O'Sullivan, B. (eds.) Trends in Constraint Programming, ch. 14, pp. 261–267. ISTE (2007)
10. Li, C.M., Wei, W., Zhang, H.: Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 121–133. Springer, Heidelberg (2007)
11. Morris, P.: The Breakout Method for Escaping from Local Minima. In: Proceedings of AAAI 1993, pp. 40–45. AAAI Press, Menlo Park (1993)
12. Pham, D.N., Gretton, C.: gnovelty+, `http://www.satcompetition.org/2007/contestants.html`
13. Prestwich, S.: Random Walk with Continuously Smoothed Variable Weights. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 203–215. Springer, Heidelberg (2005)
14. Selman, B., Kautz, H., Cohen, B.: Noise Strategies for Improving Local Search. In: Proceedings of AAAI 1994, pp. 337–343. AAAI Press, Menlo Park (1994)
15. Wei, W., Li, C.M., Zhang, H.: Deterministic and Random Selection of Variables in Local Search for SAT, `http://www.satcompetition.org/2007/contestants.html`
16. Wei, W., Li, C.M., Zhang, H.: Criterion for Intensification and Diversification in Local Search for SAT. In: Proceedings of LSCS 2007, pp. 2–16 (2007)
17. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla-07: The Design and Analysis of an Algorithm Portfolio for SAT. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 712–727. Springer, Heidelberg (2007)