Journal on Satisfiability, Boolean Modeling and Computation 4 (2008) 219-237

A Switching Criterion for Intensification and Diversification in Local Search for SAT*, †

Wanxia Wei wanxia.wei@unb.ca
Faculty of Computer Science,
University of New Brunswick

Chu Min Li chu-min.li@u-picardie.fr
MIS,
Université de Picardie Jules Verne

Harry Zhang hzhang@unb.ca
Faculty of Computer Science,
University of New Brunswick

Abstract

We propose a new switching criterion, namely the evenness or unevenness of the distribution of variable weights, and use this criterion to combine intensification and diversification in local search for SAT. We refer to the ways in which state-of-the-art local search algorithms $adaptG^2WSAT_P$ and VW select a variable to flip, as *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW, respectively. To evaluate the effectiveness of this criterion, we apply it to *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW, in which the former intensifies the search better than the latter, and the latter diversifies the search better than the former. The resulting local search algorithm, which switches between *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW in every step according to this criterion, is called *Hybrid*. Our experimental results show that, on a broad range of SAT instances presented in this paper, *Hybrid* inherits the strengths of $adaptG^2WSAT_P$ and VW, and exhibits generally better performance than $adaptG^2WSAT_P$ and VW. In addition, *Hybrid* compares favorably with state-of-the-art local search algorithm *R+adaptNovelty+* on these instances. Furthermore, without any manual tuning parameters, *Hybrid* solves each of these instances in a reasonable time, while $adaptG^2WSAT_P$, VW, and *R+adaptNovelty+* have difficulty on some of these instances.

Keywords: *SAT, local search, switching criterion, intensification, diversification, distribution of variable weights*

Submitted October 2007; revised March 2008; published June 2008

1. Introduction

Intensification and diversification are two properties of a search process. 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

∗ A preliminary version of this paper was presented at the 4th International Workshop on LSCS [21].

† 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).

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].

There appear to be two classes of local search algorithms, those that intensify the search well, and those that diversify the search well. The first class of algorithms includes $GSAT$ [18], $HSAT$ [2], $WalkSAT$ [17], $R{+}adaptNovelty{+}$ [1], G^2WSAT [7], and $adaptG^2WSAT_P$ [8, 9]. Among these algorithms, $R{+}adaptNovelty{+}$ integrates restricted resolution in a preprocessing phase into $AdaptNovelty{+}$ [4], G^2WSAT deterministically uses promising decreasing variables, and $adaptG^2WSAT_P$ implements the adaptive noise mechanism from [4] in G^2WSAT and contains limited look-ahead moves. The second class of algorithms includes the variable weighting algorithm VW [15], which uses variable weights to diversify the search. This second class of algorithms also includes clause weighting algorithms, such as $Breakout$ [14], DLM (Discrete Lagrangian Method) [22], Guided Local Search (GLSSAT) [13], SDF (Smoothed Descent and Flood) [16], $SAPS$ (Scaling And Probabilistic Smoothing) [6], $RSAPS$ (Reactive SAPS) [6], and $PAWS$ (Pure Additive Weighting Scheme) [19], because according to [20], clause weighting works as a form of diversification.

$R{+}adaptNovelty{+}$, G^2WSAT with noise p=0.50 and diversification probability dp=0.05, and VW won the gold, silver, and bronze medals, respectively, in the satisfiable random formula category in the SAT 2005 competition.[1.] Experiments in [8, 9] show that, without any manual noise or other parameter tuning, $adaptG^2WSAT_P$ shows generally good performance, compared with G^2WSAT with optimal static noise settings, or is sometimes even better than G^2WSAT, and that $adaptG^2WSAT_P$ compares favorably with $R{+}adaptNovelty{+}$ and VW.

Nevertheless, each local search algorithm or heuristic has weaknesses. To examine the weaknesses of the above two classes of algorithms, we conduct experiments with one state-of-the-art algorithm from each class. The algorithm from the first class is $adaptG^2WSAT_P$, and the algorithm from the second class is VW. Our experimental results show that the performance of $adaptG^2WSAT_P$ is poor on some instances for which a local search algorithm may result in imbalanced flip numbers of variables, and that the performance of VW is poor on some instances for which a local search algorithm may result in balanced flip numbers of variables. The poor performance of $adaptG^2WSAT_P$ may result from the fact that this algorithm does not employ any weighting to diversify the search. The poor performance of VW may result from the fact that VW always considers variable weights to diversify the search when choosing a variable to flip, even if the flip numbers of variables are balanced. In fact, when the flip numbers of variables are balanced, i.e., when searches by VW are diversified, VW should intensify the search well.

In the literature, several local search algorithms switch between heuristics [3, 11, 7, 8, 9]. $UnitWalk$ [3] combines unit clause elimination and local search. $UnitWalk$ 0.98, one of the latest versions of $UnitWalk$, alternates between $WalkSAT$-like and $UnitWalk$-like searches. $QingTing2$ [11] switches between $WalkSAT$ [17] and $QingTing1$, which implements $UnitWalk$ with a new unit-propagation technique. G^2WSAT [7] switches between a variant of $GSAT$ and $Novelty{+}{+}$. The local search algorithm $adaptG^2WSAT_P$ [8, 9] switches between a variant of $GSAT$ and $Novelty{+}{+}_P$. However, none of these algorithms

1. http://www.satcompetition.org/

switches from one heuristic to another during the search to diversify the search by using variable weighting.

In this paper, we propose a new switching criterion: the evenness or unevenness of the distribution of variable weights. We refer to the ways in which local search algorithms $adaptG^2WSAT_P$ and VW select a variable to flip, as *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW, respectively. Then, to evaluate the effectiveness of this switching criterion, we develop a new local search algorithm called $Hybrid$, which switches between *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW in every step according to this switching criterion. This new algorithm allows suitable diversification strategies to complement intensification strategies by switching between *heuristic* $adaptG^2WSAT_P$ and *heuristic* VW. Our experimental results show that, on a broad range of SAT instances presented in this paper, $Hybrid$ inherits the strengths of $adaptG^2WSAT_P$ and VW.

2. Review of Algorithms $adaptG^2WSAT_P$ and VW

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 difference between $make(x)$ and $break(x)$. Let *best* and *second* be the best and second best variables in a randomly selected unsatisfied clause c according to their scores. Heuristic $Novelty$ [12] selects a variable to flip from c as follows.

$Novelty(p)$: If *best* is not the most recently flipped variable in c, then pick it. Otherwise, with probability p, pick *second*, and with probability 1-p, pick *best*.

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 [7] 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 variables, $x \neq y$, and x be not decreasing with respect to A. If $score_C(x) > 0$ where C is the new assignment after flipping y, 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.

According to the above definition of promising decreasing variables, flipping such a variable not only decreases the number of unsatisfied clauses but also probably allows local search to explore new promising regions in the search space.

Let assignment B be obtained from A by flipping x, and let x' be the best promising decreasing variable with respect to B. The promising score of x with respect to A, $pscore_A(x)$, is defined in [8, 9, 10] 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.[2]

If there are promising decreasing variables with respect to B, $pscore_A(x)$ 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,

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

Heuristic $Novelty{+}{+}_P$ [8, 9] selects a variable to flip from c as follows.

$Novelty{+}{+}_P(p, dp)$: With probability dp (diversification probability), flip a variable in c whose flip falsifies the least recently satisfied clause. With probability 1-dp, do as $Novelty$, but flip *second* if *best* is more recently flipped than *second* and if $pscore(second) \geq pscore(best)$.

If promising decreasing variables exist, the local search algorithm $adaptG^2WSAT_P$ [8, 9] flips the promising decreasing variable with the largest computed promising score. Otherwise, $adaptG^2WSAT_P$ selects a variable to flip from a randomly chosen unsatisfied clause using $Novelty{+}{+}_P$. We refer to the way in which the algorithm $adaptG^2WSAT_P$ selects a variable to flip, as *heuristic* $adaptG^2WSAT_P$.

The local search algorithm VW [15] uses variable weights to diversify the search. This algorithm initializes the weight of a variable x, $var\_weight[x]$, to 0 and updates and smoothes $var\_weight[x]$ each time x is flipped, using the following formula:

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

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.

Clause weighting algorithms usually use expensive smoothing phases in which all clause weights are adjusted to reduce the differences between them. In contrast, VW uses an efficient variable weight smoothing technique, namely continuous smoothing, in which smoothing occurs as weights are updated. We describe this continuous smoothing in the following. In Formula 1, there are two extreme values for parameter s. The first one is s = 1, and this value causes variables to forget their flip histories. That is, only the most recent flip of a variable affects the weight of this variable. The second one is $s = 0$. This value causes the weight of a variable to behave like a simple counter of the flips of this variable, so every flip of a variable has an equal effect on the weight of this variable. VW adjusts s during the search and lets s be a value between these two extreme values, i.e., $0 < s < 1$. When $0 < s < 1$, older events in the search history have lesser but non-zero effects on variable weights.

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 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. We call this rule the *low variable weight favoring rule*,

2. x' has the highest $score_B(x')$ among all promising decreasing variables with respect to B.
3. A freebie variable is a variable with a break of 0.

and describe it as follows. Let the best variable in a randomly selected unsatisfied clause c so far be *best*. If a variable x in c has fewer breaks than *best*, x becomes the new *best*. If x has the same number of breaks as *best* but a lower variable weight, x becomes the new *best*. If x has more breaks than *best* but a lower variable weight, x becomes the new *best* with a probability that is equal to or higher than $1/2^{break_x - break_{best}}$ where $break_x$ and $break_{best}$ are the breaks of x and *best*, respectively. We refer to the way in which the algorithm VW selects a variable to flip, as *heuristic* VW.

3. Motivation

We observe that searches by VW are better diversified than searches by $adaptG^2WSAT_P$, and that searches by $adaptG^2WSAT_P$ are better intensified than searches by VW. In addition, we conjecture that variable weights provide meaningful information for VW to diversify the search, usually when the flip numbers of variables are imbalanced, and that $adaptG^2WSAT_P$ intensifies the search well, usually when the flip numbers of variables are generally balanced. To empirically confirm our observations and empirically verify our conjectures, we conduct experiments with VW and $adaptG^2WSAT_P$.

We make $adaptG^2WSAT_P$ calculate variable weights in the same way as does VW, although $adaptG^2WSAT_P$ does not consider variable weights when choosing a variable to flip. We run VW and $adaptG^2WSAT_P$ on two classes of instances.[4.] The source code of VW was obtained from the organizer of the SAT 2005 competition. The first class comes from the SAT 2005 competition benchmark[5.] and includes the 8 random instances from O*1582 to O*1589. The second class is from Miroslav Velev's SAT Benchmarks[6.] and consists of all of the formulas from Superscalar Suite 1.0a (SSS.1.0a) except for *bug54.[7.] Each algorithm is run 100 times ($Maxtries = 100$). The cutoffs are set to 10^8 ($Maxsteps = 10^8$) and 10^7 ($Maxsteps = 10^7$) for a random instance and an instance from SSS.1.0a, respectively.

"Depth" is one of the three measures introduced in [16] and assesses how many clauses remain unsatisfied during the search. We make VW and $adaptG^2WSAT_P$ calculate the average depth (the number of unsatisfied clauses), the average coefficient of variation of distribution of variable weights (coefficient of variation = standard deviation / mean value), and the average division of the maximum variable weight by the average variable weight, over all search steps. In Tables 1 and 2, we report the calculated average depth ("depth"), the calculated average coefficient of variation of distribution of variable weights ("cv"), and the calculated average division of maximum variable weight by average variable weight ("div"), each value being averaged over 100 runs ($Maxtries = 100$). A run is successful if it finds a solution within a cutoff ($Maxsteps$). The success rate of an algorithm for an instance is the number of successful runs divided by the value of $Maxtries$. In these tables, we also report success rates ("suc"). In addition, in the last row of each table, we present the average of the values in each column ("avg").

4. All experiments reported are conducted in Chorus, which consists of 2 dual processor master nodes with hyperthreading enabled and 80 dual processor compute nodes. Each compute node has two 2.8GHz Intel Xeon processors with 2 to 3 Gigabytes of memory.
5. http://www.lri.fr/~simon/contest/results/
6. http://www.ece.cmu.edu/~mvelev/sat_benchmarks.html
7. The instance *bug54 is hard for every algorithm discussed in this paper. For example, if we run VW on *bug54 ($Maxsteps = 10^8$), the success rate is only 0.40%.

Table 1. Performance and distributions of variable weights for VW and $adaptG^2WSAT_P$ on the 8 random instances.

	VW				$adaptG^2WSAT_P$			
	depth	cv	div	suc	depth	cv	div	suc
O*1582	23.22	0.000	1.000	0.30	10.30	0.010	1.017	1.00
O*1583	22.68	0.001	1.001	0.69	10.17	0.018	1.052	1.00
O*1584	23.24	0.000	1.002	0.38	10.27	0.009	1.027	1.00
O*1585	23.19	0.000	1.001	0.35	10.39	0.008	1.015	1.00
O*1586	22.21	0.000	1.001	0.25	9.73	0.005	1.016	1.00
O*1587	22.66	0.001	1.002	0.94	9.98	0.032	1.277	1.00
O*1588	22.75	0.000	1.000	0.30	10.02	0.007	1.017	0.99
O*1589	22.57	0.000	1.000	0.40	10.11	0.009	1.068	1.00
avg	22.82	0.000	1.001	0.45	10.12	0.012	1.061	1.00

Table 1 shows that on the random instances, the average depths of VW and $adaptG^2WSAT_P$ are 22.82 and 10.12, respectively, and that on these instances, the average coefficients of variation of VW and $adaptG^2WSAT_P$ are 0.000 and 0.012, respectively. On these random instances, the average success rate of VW is 0.45, while that of $adaptG^2WSAT_P$ is 1.00. Table 2 shows that on the instances from SSS.1.0a, the average depths of VW and $adaptG^2WSAT_P$ are 84.59 and 10.13, respectively, and that on these instances, the average coefficients of variation of VW and $adaptG^2WSAT_P$ are 1.820 and 10.204, respectively. On the instances from SSS.1.0a, the average success rate of VW is 1.00, while that of $adaptG^2WSAT_P$ is 0.23. That is, regardless of the performance of VW and $adaptG^2WSAT_P$, the average coefficient of variation of $adaptG^2WSAT_P$ is significantly higher than that of VW, and the average depth of $adaptG^2WSAT_P$ is significantly lower than that of VW.

Table 2. Performance and distributions of variable weights for VW and $adaptG^2WSAT_P$ on the 8 instances in SSS.1.0a.

	VW				$adaptG^2WSAT_P$			
	depth	cv	div	suc	depth	cv	div	suc
*bug3	7.36	0.872	3.979	0.97	4.25	11.584	203.114	0.00
*bug4	28.05	1.685	10.144	1.00	4.68	10.793	158.692	0.04
*bug5	26.92	1.511	8.702	1.00	4.94	11.810	190.262	0.03
*bug17	288.18	2.727	29.564	1.00	23.92	7.722	161.185	0.64
*bug38	52.74	1.684	10.501	1.00	5.57	11.734	208.653	0.11
*bug39	53.41	1.836	13.466	1.00	12.50	8.930	139.881	0.41
*bug40	74.62	1.899	15.235	1.00	7.04	10.618	178.253	0.14
*bug59	145.43	2.342	22.812	1.00	18.13	8.443	123.465	0.49
avg	84.59	1.820	14.300	1.00	10.13	10.204	170.438	0.23

The lower the average depth is, the fewer the unsatisfied clauses are, and the better intensified the search is. The distribution of variable weights reflects the flipping history of variables. If all variables have roughly equal chances of being flipped, all variables should have approximately equal weights, and the coefficient of variation of the distribution of variable weights should be low. Conversely, if some variables have been flipped much more frequently than others, the weights of these variables should be much higher than those of others, and the coefficient of variation of the distribution of variable weights should be high. That is, the higher the average coefficient of variation is, the more variable weights far from the mean value exist, the more imbalanced variable weights are, and the less well diversified the search is. Thus, the results in Tables 1 and 2 confirm that, regardless of the performance of VW and *adaptG*$^2WSAT_P$, VW can diversify the search better than *adaptG*$^2WSAT_P$, and *adaptG*$^2WSAT_P$ can intensify the search better than VW.

According to Table 1, on the random instances, the average coefficients of variation of VW and *adaptG*$^2WSAT_P$ are 0.000 and 0.012, respectively. As indicated in Table 2, on the instances from SSS.1.0a, the average coefficients of variation of VW and *adaptG*$^2WSAT_P$ are 1.820 and 10.204, respectively. That is, the random instances usually result in balanced variable weights while the instances from SSS.1.0a usually result in unbalanced variable weights. As shown in Table 1, on the random instances, the average success rate of VW is 0.45, while that of *adaptG*$^2WSAT_P$ is 1.00. Hence, the results in these two tables suggest that an algorithm should not consider variable weights when selecting a variable to flip if the distribution of variable weights is balanced. Instead, an algorithm should ignore variable weights and concentrate on improving the objective function to intensify the search well. As shown in in Table 2, on the instances from SSS.1.0a, the average success rate of VW is 1.00, while that of *adaptG*$^2WSAT_P$ is 0.23. Thus, the results in these two tables also suggest that an algorithm should make use of variable weights to diversify the search well when the distribution of variable weights is imbalanced.

As indicated in Table 1, on the random instances, the averages of the values for *div* in VW and *adaptG*$^2WSAT_P$ are 1.001 and 1.061, respectively, while as indicated in Table 2, on the instances from SSS.1.0a, the averages of the values for *div* in VW and *adaptG*$^2WSAT_P$ are 14.300 and 170.438, respectively. That is, the maximum variable weight on the instances from SSS.1.0a usually deviates from the average variable weight to a greater degree than does the maximum variable weight on the random instances. Therefore, the results in these two tables suggest that, similar to the coefficient of variation of distribution of variable weights, the division of the maximum variable weight by the average variable weight also indicates whether variable weights are balanced. In fact, calculating the division is not time-consuming, but calculating the coefficient of variation is.

4. A New Switching Criterion

In this section, we propose a new switching criterion: the evenness or unevenness of the distribution of variable weights. Additionally, we propose a switching strategy that uses this switching criterion. Furthermore, we introduce a new local search algorithm $Hybrid$ that implements this proposed switching strategy.

4.1 Evenness or Unevenness of Distribution of Variable Weights

We propose a new switching criterion: the evenness or unevenness of the distribution of variable weights. Assume that variable weights are updated using Formula 1. Assume that γ is an integer and $\gamma > 1$. If the maximum weight is at least γ times as high as the average weight, the distribution of variable weights is considered *uneven* and the step is called *an* *uneven step*. Otherwise, the distribution is considered *even* and the step is called *an even step*. We use an uneven or even distribution of variable weights as a means to determine whether or not a search is undiversified in a step. More specifically, an uneven distribution and an even distribution of variable weights correspond to an undiversified search and a diversified search, respectively, in a step.

One switching strategy that is based on this switching criterion is as follows. In each search step, if the distribution of variable weights is uneven, i.e., if a search is not diversified, a heuristic that can diversify the search well is used to choose a variable to flip. In each search step, if the distribution of variable weights is even, i.e., if a search is diversified, a heuristic that can intensify the search well is used to choose a variable to flip.

We compare the above switching strategy with those used in $QingTing2$ [11], $UnitWalk$ 0.98 [3], G^2WSAT [7], and $adaptG^2WSAT_P$ [8, 9]. Before solving an instance, $QingTing2$ samples this instance for a fixed number of trials. During each trial, $QingTing2$ starts by assigning a random value to an unassigned variable chosen at random. This step is called a random assignment. $QingTing2$ then propagates this randomly assigned value through unit propagation. When the unit propagation stops, $QingTing2$ conducts another random assignment. Such a process repeats until all the clauses in the formula of this instance are either conflicted or satisfied. In [11], variable immunity is defined as the ratio of the number of random assignments in a trial to the number of variables of an instance. Intuitively, the higher a variable immunity is, the less dependence the variables of an instance have. Then, for this instance, according to whether the obtained variable immunity is higher than a threshold, $QingTing2$ decides to use either $WalkSAT$ or $QingTing1$. During the search, for this instance, $QingTing2$ never switches to the other heuristic. Let n be the number of variables of an instance. $UnitWalk$ 0.98 repeats periods[8.] of $UnitWalk$ until the following two conditions hold: k opposite unit clause pairs are found during a period and k' of these pairs are found in the previous period, where k and k' are integers, $k \geq n/12$, and $k \geq k'$. When these two conditions hold, $UnitWalk$ 0.98 switches to $WalkSAT$, for which the cutoff is set to $n^2/2$. During the search, both G^2WSAT and $adaptG^2WSAT_P$ switch between heuristics according to whether there are promising decreasing variables. When the distribution of variable weights is uneven, our proposed switching strategy uses a heuristic that can diversify the search well to choose a variable to flip. Otherwise, this switching strategy uses a heuristic that can intensify the search well to choose a variable to flip.

In summary, our proposed switching strategy has two features. First, it diversifies the search when the distribution of variable weights is uneven, and intensifies the search when the distribution of variable weights is even, while none of the strategies used in $QingTing2$, $UnitWalk$ 0.98, G^2WSAT, and $adaptG^2WSAT_P$ has these functions. Second, like those used in G^2WSAT and $adaptG^2WSAT_P$, it considers whether to switch to the other heuristic in every step, while those used in $QingTing2$ and $UnitWalk$ 0.98 do not.

4.2 Algorithm $Hybrid$

To evaluate the effectiveness of the proposed switching criterion, we implement the proposed switching strategy in an algorithm called $Hybrid$, which is described in Fig. 1. In each step, $Hybrid$ chooses a variable to flip according to *heuristic VW* if the distribution of variable

8. An iteration of the outer loop of $UnitWalk$ is called a period.

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

$A \leftarrow$ randomly generated truth assignment;
for each variable x **do** initialize $flip\_time[x]$ and $var\_weight[x]$ to 0;
initialize p, dp, max_weight, and ave_weight to 0;
store promising decreasing variables in stack *DecVar*;
for *flip*$\leftarrow$1 **to** *Maxsteps* **do**
    if A satisfies $\mathcal{F}$ **then return** A;
    if $max\_weight \geq \gamma \times ave\_weight$
    then $y \leftarrow$*heuristic* $VW(p)$;
    else $y \leftarrow$*heuristic adaptG*$^2WSAT_P(p, dp)$;
    $A \leftarrow A$ with y flipped; adapt p and dp;
    update $flip\_time[y]$, $var\_weight[y]$, max_weight, ave_weight, and *DecVar*;
return Solution not found;

Figure 1. Algorithm *Hybrid*

weights is uneven, and selects a variable to flip according to *heuristic adaptG*$^2WSAT_P$ otherwise. As a result, *Hybrid* combines intensification strategies with suitable diversification strategies by switching between these two heuristics.

Hybrid uses a quite simple switching strategy to measure whether a search is diversified. Alternative switching strategies can be based on mobility and coverage, the other two measures proposed in [16], which determine how rapidly and systematically, respectively, the search explores the entire space. These two measures were introduced to deal with the following situation: a local search algorithm achieves a good depth value but easily gets stuck in local minima if it fails to explore the search space rapidly and systematically. Our prospective research will involve using mobility and coverage to establish new switching criteria to measure whether a search is diversified. Compared with these alternative switching strategies, the simple switching strategy that *Hybrid* uses is not time-consuming when implemented. Though this simple strategy is easy and fast to implement, according to our experimental results presented in Section 5, this strategy is effective.

Hybrid is an example that uses the proposed switching criterion. This switching criterion can be used in other local search algorithms that combine intensification strategies with diversification strategies.

5. Evaluation

We define the switching criterion used in *Hybrid* more specifically in this section than in Section 4.1. In addition, we compare the performance of *Hybrid* with those of state-of-the-art local search algorithms such as *adaptG*$^2WSAT_P$, VW, and *R+adaptNovelty+* on a wide range of SAT instances. Moreover, we justify the proposed switching strategy used in *Hybrid*.

5.1 Groups of Instances

We conduct experiments on 11 groups of benchmark SAT problems (65 problems). Structured problems come from the SATLIB repository[9.] and Miroslav Velev's SAT Benchmarks. These problems include bw_large.c and bw_large.d in blocksworld, e0ddr2*1, e0ddr2*4, enddr2*1, enddr2*8, ewddr2*1, and ewddr2*8 in Beijing, g250.29 in GCP, logi*.c in logistics, par16-1, par16-2, par16-3, par16-4, and par16-5 in parity, the 10 satisfiable instances in QG,[10.] and all satisfiable formulas in SSS.1.0a except for *bug54. Crafted and industrial problems come from the SAT 2005 competition benchmark. Crafted problems consist of the 8 instances from g*1334 to g*1341. Industrial problems include v*1912, v*1915, v*1923, v*1924, v*1944, v*1955, v*1956, and v*1959. Random problems constitute two groups. The first group consists of the 8 instances unif04-52, unif04-62, unif04-65, unif04-80, unif04-83, unif04-86, unif04-91, and unif04-99, from the SAT 2004 competition benchmark.[11.] The second group includes the 8 instances from O*1582 to O*1589 from the SAT 2005 competition benchmark.

We select the above 11 groups of benchmark SAT problems using the following three rules. First, these problems should include those widely used benchmark problems in the literature. As a result, these problems include the entire set of instances that were used to originally evaluate $R{+}adaptNovelty{+}$ [1], the best local search algorithm in the SAT 2005 competition. Second, these problems should constitute structured, crafted, industrial, and random instances. Third, these problems should include those instances that, for *Hybrid*, usually lead to the following two combinations of the distributions of variable weights: the distributions of variable weights are even and the distributions of variable weights are uneven. Specifically, among these 65 instances, for *Hybrid*, the instances in parity and the 8 random instances from O*1582 to O*1589 generally result in even distributions of variable weights. The instances from Beijing and from SSS.1.0a, and the crafted instances from g*1334 to g*1341 usually lead to uneven distributions of variable weights.

The cutoff ($Maxsteps$) is set to 10^6 for the instance in logistics, to 10^7 for all instances in blocksworld, Beijing, GCP, SSS.1.0a, and the group from the SAT 2004 competition, to 10^8 for the crafted instances, the industrial instances, and the random instances from the SAT 2005 competition benchmark, and to 10^9 for all instances in parity. $Maxsteps$ is set to 10^8 for qg7-13 in QG and to 10^7 for the other instances in this group. Each instance is executed 250 times ($Maxtries = 250$). The cutoff for each instance is set to a fixed value, to ensure that at least one algorithm discussed achieves a success rate greater than 50% in order to calculate median flip number and median run time based on these 250 runs. We report success rate ("suc"), median flip number ("#flips"), and median run time ("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 flip number and the median run time, respectively. Results in bold indicate the best performance for an instance.

9. http://www.satlib.org/
10. Since these QG instances contain unit clauses, we simplify them using *my_compact*, which was downloaded from http://www.laria.u-picardie.fr/~cli.
11. http://www.lri.fr/~simon/contest04/results/

5.2 Updating Variable Weights and Defining Switching Criterion

Like VW, $Hybrid$ updates variable weights using Formula 1. To adapt to $Hybrid$, parameter s in this formula is fixed to 0. That is, in $Hybrid$, $s = 0$. When s is 0, the weight of a variable defined in this formula is just a counter of the number of flips of this variable. In contrast, s in VW is adjusted during the search ($s > 0$).

The higher parameter γ in $Hybrid$ is, the fewer the uneven steps exist, and the less frequently $Hybrid$ chooses *heuristic* VW to select a variable to flip. We run different versions of $Hybrid$ with γ=4, 10, 15, 20, 25, 30, 35, 40, and 45. Our experimental results show that on the hardest instances from the 11 groups, $Hybrid$ with γ=10 exhibits the best overall performance among all of these versions. So, in $Hybrid$, the default value of γ is set to 10.

Table 3. Experimental results for $adaptG^2WSAT_P$, VW, $Hybrid\_4$, $Hybrid$ (γ=10), and $Hybrid\_45$ on the hardest instances from the first category. In $Hybrid\_4$, $Hybrid$ (γ=10), and $Hybrid\_45$, $s = 0$.

		adapt∗	*VW*	*Hybrid_4*	*Hybrid*	*Hybrid_45*
g250.29	r_unev			99.98%	59.17%	21.25%
	#flips	637472	*$> 10^7$*	*$> 10^7$*	1306322	**590009**
	time	28.2	*n/a*	*n/a*	94.0	**26.3**
	suc	**1.00**	0.18	*0.00*	0.88	**1.00**
par16-2	r_unev			21.86%	1.00%	0.00%
	#flips	**106070896**	*$> 10^9$*	867375405	152549064	109877709
	time	**57.0**	*n/a*	540.5	92.5	109.5
	suc	**1.00**	*0.00*	0.54	**1.00**	**1.00**
v*1915	r_unev			0.50%	0.14%	0.02%
	#flips	11570303	*$> 10^8$*	**10165555**	11904448	11261618
	time	**372.6**	*n/a*	416.5	399.1	422.5
	suc	**1.00**	*0.18*	**1.00**	**1.00**	0.99
unif04-83	r_unev			25.89%	3.09%	0.01%
	#flips	5260203	*$> 10^7$*	5856586	5827928	**4482255**
	time	6.3	*n/a*	7.9	8.8	**6.0**
	suc	**0.77**	*0.24*	0.66	0.74	**0.77**
O*1586	r_unev			0.04%	0.01%	0.00%
	#flips	15649195	*$> 10^8$*	15169011	**14538393**	14782108
	time	**225.5**	*n/a*	233.8	249.3	230.6
	suc	**0.99**	*0.27*	0.98	**0.99**	**0.99**

Tables 3, 4, and 5 compare the performance of $Hybrid$ (γ=10), $Hybrid\_4$ ($Hybrid$ with γ=4), and $Hybrid\_45$ ($Hybrid$ with γ=45) on the hardest instances from the 11 groups.[12] In these tables, we also report the ratio of uneven steps to total steps ("r_unev"), which is averaged over 250 runs. This ratio is also the ratio of steps in which *heuristic* VW is used to select a variable to flip, to all steps. In addition, we report success rate ("suc") in these tables. We group these instances into three categories: those that are hard for the algorithm VW but are not hard for the algorithm $adaptG^2WSAT_P$, those that are not hard for the algorithm VW but are hard for the algorithm $adaptG^2WSAT_P$, and those that are not hard for either algorithm. For the first category, which includes g250.29, par16-2, v*1915, unif04-83, and O*1586, r_unev in $Hybrid$ (γ=10) is generally lower than 50%. As a result, in most steps, $Hybrid$ (γ=10) usually chooses *heuristic* $adaptG^2WSAT_P$ to

12. In these three tables, *adapt∗* and $Hybrid$ refer to $adaptG^2WSAT_P$ and $Hybrid$ (γ=10), respectively. Results in italics indicate the poorest performance for an instance.

Table 4. Experimental results for *adaptG*^{2}*WSAT*$_P$, *VW*, *Hybrid_4*, *Hybrid* (γ=10), and *Hybrid_45* on the hardest instances from the second category. In *Hybrid_4*, *Hybrid* (γ=10), and *Hybrid_45*, $s = 0$.

		adapt∗	*VW*	*Hybrid_4*	*Hybrid*	*Hybrid_45*
qg7-13	r_unev			64.48%	42.53%	24.53%
	#flips	$> 10^8$	8843466	2581390	**1881094**	$> 10^8$
	time	*n/a*	307.6	151.5	**32.3**	*n/a*
	suc	0.48	**0.90**	0.76	0.71	*0.44*
*bug3	r_unev			99.98%	85.77%	3.28%
	#flips	$> 10^7$	1786329	635297	**628668**	$> 10^7$
	time	*n/a*	3.7	3.1	**3.0**	*n/a*
	suc	*0.00*	**0.98**	0.96	0.97	0.34
g*1341	r_unev			96.79%	87.38%	42.81%
	#flips	$> 10^8$	6253863	**2683350**	2751076	2985281
	time	*n/a*	**17.8**	20.1	23.1	40.2
	suc	*0.00*	**1.00**	**1.00**	**1.00**	**1.00**

select a variable to flip. Conversely, for g250.29, r_unev in *Hybrid_4* is too high, as high as 99.98%, resulting in the poor performance of *Hybrid_4* on this instance. For the second category, which consists of qg7-13, *bug3, and g*1341, r_unev in *Hybrid* (γ=10) is generally higher than 50%. Consequently, in most steps, *Hybrid* (γ=10) usually chooses *heuristic VW* to select a variable to flip. By contrast, for qg7-13 and *bug3 , the values of r_unev in *Hybrid_45* are too low, as low as 24.53% and 3.28%, respectively, leading to the poor performance of *Hybrid_45* on these two instances. For the third category, which includes bw_large.d, e0ddr2*1, and logi*.c, r_unev in *Hybrid* (γ=10) can be lower or higher than 50%. Therefore, the success of searches by *Hybrid* for an instance lies in whether, based on the switching criterion, in most search steps, *Hybrid* usually chooses the appropriate heuristic to select a variable to flip for this instance.

Table 5. Experimental results for *adaptG*^{2}*WSAT*$_P$, *VW*, *Hybrid_4*, *Hybrid* (γ=10), and *Hybrid_45* on the hardest instances from the third category. In *Hybrid_4*, *Hybrid* (γ=10), and *Hybrid_45*, $s = 0$.

		adapt∗	*VW*	*Hybrid_4*	*Hybrid*	*Hybrid_45*
bw_large.d	r_unev			99.89%	99.54%	18.66%
	#flips	2124858	*2963500*	**822900**	962677	1984479
	time	12.4	18.1	**4.8**	6.3	*22.5*
	suc	*0.96*	**0.98**	0.97	**0.98**	**0.98**
e0ddr2*1	r_unev			100%	100%	96.00%
	#flips	3068450	*6549282*	**105122**	114774	211200
	time	15.3	*22.5*	**2.7**	2.9	4.2
	suc	0.99	*0.66*	**1.00**	**1.00**	**1.00**
logi*.c	r_unev			99.21%	67.42%	0.23%
	#flips	49469	*70446*	**17602**	19038	47086
	time	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
	suc	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

We allow *Hybrid* to adjust s in the same way as does *VW* ($s > 0$), and we call this version of *Hybrid* *H_asVW*. Our experimental results show that *Hybrid* (γ=10) exhibits better overall performance than *H_asVW* (γ=10) on the hardest instances from the 11 groups. Table 6 presents the performance of these two algorithms on these instances.

Table 6. Experimental results for H_asVW (γ=10) and $Hybrid$ (γ=10) on the hardest instances from the 11 groups. In H_asVW (γ=10), $s > 0$, while in $Hybrid$ (γ=10), $s = 0$.

	H_asVW			$Hybrid$		
	#flips	time	suc	#flips	time	suc
bw_large.d	1881240	15.9	**0.98**	**962677**	**6.3**	**0.98**
e0ddr2*1	240223	4.0	**1.00**	**114774**	**2.9**	**1.00**
g250.29	**689661**	**39.0**	**1.00**	1306322	94.0	0.88
logi*.c	37765	**0.1**	**1.00**	**19038**	**0.1**	**1.00**
par16-2	**99909500**	**60.8**	**1.00**	152549064	92.5	**1.00**
qg7-13	$> 10^8$	n/a	0.44	**1881094**	**32.3**	**0.71**
*bug3	$> 10^7$	n/a	0.14	**628668**	**3.0**	**0.97**
g*1341	6831055	41.2	0.98	**2751076**	**23.1**	**1.00**
v*1915	**10477276**	**358.8**	**1.00**	11904448	399.1	**1.00**
unif04-83	**4421929**	**6.0**	**0.77**	5827928	8.8	0.74
O*1586	15271672	263.0	**0.99**	**14538393**	**249.3**	**0.99**

Table 7. Experimental results for R+$adaptNovelty$+, $adaptG^2WSAT_P$, VW, and $Hybrid$ (γ=10) on the structured and crafted instances. In $Hybrid$ (γ=10), $s = 0$.

	R+$adaptNovelty$+			$adaptG^3WSAT_P$			VW			$Hybrid$ (γ=10)		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
bw_large.c†	9489817	29.1	0.52	992093	3.5	**1.00**	1868393	6.0	**1.00**	**597473**	**2.5**	0.99
bw_large.d	$> 10^7$	n/a	0.29	2124858	12.4	0.96	2963500	18.1	**0.98**	**962677**	**6.3**	**0.98**
e0ddr2*1†	2488226	10.6	0.92	3068450	15.3	0.99	6549282	22.5	0.66	**114774**	**2.9**	**1.00**
e0ddr2*4†	355044	**1.5**	**1.00**	694059	4.3	**1.00**	1894243	7.9	0.98	**71214**	2.7	**1.00**
enddr2*1†	331420	**1.6**	**1.00**	641226	4.3	**1.00**	4484178	17.6	0.83	**54245**	2.6	**1.00**
enddr2*8†	**11753**	**0.0**	**1.00**	555475	3.8	**1.00**	3398071	16.1	0.92	47090	2.7	**1.00**
ewddr2*1†	154825	**0.7**	**1.00**	520705	3.6	**1.00**	4052096	16.5	0.88	**42881**	2.5	**1.00**
ewddr2*8†	**32527**	**0.1**	**1.00**	432671	3.1	**1.00**	4608302	20.5	0.86	35079	2.4	**1.00**
g250.29	733420	**23.2**	**1.00**	**637472**	28.2	**1.00**	$> 10^7$	n/a	0.18	1306322	94.0	0.88
logi*.c†	57693	**0.1**	**1.00**	49469	**0.1**	**1.00**	70446	**0.1**	**1.00**	**19038**	**0.1**	**1.00**
par16-1†	80339283	37.6	**1.00**	**55017679**	**28.0**	**1.00**	$> 10^9$	n/a	0.00	65354529	37.8	**1.00**
par16-2†	324826713	157.5	0.89	**106070896**	**57.0**	**1.00**	$> 10^9$	n/a	0.00	152549064	92.5	**1.00**
par16-3†	224140856	107.4	0.93	97156387	**51.6**	**1.00**	$> 10^9$	n/a	0.00	**87443760**	53.5	**1.00**
par16-4†	274054172	129.7	0.92	118557332	**61.4**	**1.00**	$> 10^9$	n/a	0.00	**108114087**	63.6	**1.00**
par16-5†	264871971	125.0	0.94	**83028280**	**44.4**	**1.00**	$> 10^9$	n/a	0.01	105083154	63.0	**1.00**
qg1-07†	9609	**0.0**	**1.00**	6206	**0.0**	**1.00**	27607	0.1	**1.00**	**5722**	**0.0**	**1.00**
qg1-08†	733077	2.5	**1.00**	**380083**	**2.3**	**1.00**	2178590	30.9	0.95	440417	3.6	**1.00**
qg2-07†	6515	**0.0**	**1.00**	4318	**0.0**	**1.00**	9847	**0.0**	**1.00**	**4226**	**0.0**	**1.00**
qg2-08†	1893442	**7.2**	0.97	1540278	10.7	**1.00**	$> 10^7$	n/a	0.48	**1353338**	10.0	**1.00**
qg3-08†	46193	**0.1**	**1.00**	35226	**0.1**	**1.00**	143567	0.2	**1.00**	**32808**	**0.1**	**1.00**
qg4-09†	119896	0.2	**1.00**	**61492**	**0.1**	**1.00**	318048	0.6	**1.00**	63634	**0.1**	**1.00**
qg5-11†	37875	**0.2**	**1.00**	20494	**0.2**	**1.00**	53713	0.5	**1.00**	**19212**	**0.2**	**1.00**
qg6-09†	638	**0.0**	**1.00**	**353**	**0.0**	**1.00**	1151	**0.0**	**1.00**	417	**0.0**	**1.00**
qg7-09†	540	**0.0**	**1.00**	**283**	**0.0**	**1.00**	1074	**0.0**	**1.00**	320	**0.0**	**1.00**
qg7-13†	5113772	66.7	0.72	$> 10^8$	n/a	0.48	8843466	307.6	**0.90**	**1881094**	**32.3**	0.71
*bug3	$> 10^7$	n/a	0.30	$> 10^7$	n/a	0.00	1786329	3.7	**0.98**	**628668**	**3.0**	0.97
*bug4	$> 10^7$	n/a	0.16	$> 10^7$	n/a	0.03	185184	**0.4**	**1.00**	**113857**	0.8	**1.00**
*bug5	$> 10^7$	n/a	0.06	$> 10^7$	n/a	0.06	280071	**0.7**	**1.00**	**102743**	0.9	**1.00**
*bug17	6420481	150.1	0.72	128497	2.6	0.70	32999	**0.3**	**1.00**	**20361**	1.4	**1.00**
*bug38	3765043	23.5	0.79	$> 10^7$	n/a	0.13	**157834**	**0.4**	**1.00**	210259	1.2	0.96
*bug39	$> 10^7$	n/a	0.46	$> 10^7$	n/a	0.39	**83287**	**0.2**	**1.00**	89306	0.7	**1.00**
*bug40	$> 10^7$	n/a	0.34	$> 10^7$	n/a	0.12	98834	**0.3**	**1.00**	**55004**	0.6	**1.00**
*bug59	387471	3.7	0.99	$> 10^7$	n/a	0.49	66090	**0.3**	**1.00**	**30058**	1.8	**1.00**
g*1334	$> 10^8$	n/a	0.07	$> 10^8$	n/a	0.10	167786	**0.2**	**1.00**	**67540**	**0.2**	**1.00**
g*1335	22181994	27.6	0.56	18665469	17.5	0.54	170227	**0.2**	**1.00**	**76130**	0.3	**1.00**
g*1336	3304773	6.1	0.72	137026	0.3	0.62	132881	**0.2**	**1.00**	**77116**	0.3	**1.00**
g*1337	41860116	60.4	0.53	94475014	100.4	0.51	304756	**0.5**	**1.00**	**142529**	0.6	**1.00**
g*1338	13007309	39.4	0.74	543830	2.2	0.56	822693	**1.9**	**1.00**	**373965**	2.0	**1.00**
g*1339	20646978	78.1	0.76	1040992	5.7	0.62	1220583	**2.9**	**1.00**	**606668**	4.4	**1.00**
g*1340	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.00	8891929	22.4	**1.00**	**2529897**	**14.0**	**1.00**
g*1341	$> 10^8$	n/a	0.00	$> 10^8$	n/a	0.00	6253863	**17.8**	**1.00**	**2751076**	23.1	**1.00**

5.3 Comparison of Performance of *Hybrid* with Performance of *adaptG*^{2}*WSAT*$_P$, *VW*, and *R+adaptNovelty+*

Table 8. Experimental results for *R+adaptNovelty+*, *adaptG*^{2}*WSAT*$_P$, *VW*, and *Hybrid* (γ=10) on the industrial and random instances. In *Hybrid* (γ=10), $s = 0$.

	R+adaptNovelty+			*adaptG*^{2}*WSAT*$_P$			*VW*			*Hybrid* (γ=10)		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
v*1912	6812718	148.7	**1.00**	**3419845**	101.6	**1.00**	61152892	3037.7	0.68	3570353	**95.6**	**1.00**
v*1915	78909897	2208.9	0.59	**11570303**	**372.6**	**1.00**	$> 10^8$	n/a	0.18	11904448	399.1	**1.00**
v*1923	2736569	51.7	**1.00**	**1300954**	31.1	**1.00**	12518563	428.5	0.99	1404437	**28.2**	**1.00**
v*1924	2931225	60.3	**1.00**	1746729	41.7	**1.00**	13744232	515.7	0.99	**1537351**	**35.0**	**1.00**
v*1944	6153990	373.9	**1.00**	3587804	221.8	**1.00**	58541545	7971.7	0.69	**3508563**	**194.1**	**1.00**
v*1955	2755333	89.5	**1.00**	1393168	65.4	**1.00**	10396220	1074.0	**1.00**	**1336078**	**50.9**	**1.00**
v*1956	2865074	114.7	**1.00**	**1494423**	**70.0**	**1.00**	13419375	1437.0	0.98	1607320	**70.0**	**1.00**
v*1959	2420412	118.3	**1.00**	559281	29.9	**1.00**	11433482	1377.2	**1.00**	**542837**	**26.4**	**1.00**
unif04-52†	$> 10^7$	n/a	0.28	4656882	**5.2**	0.79	$> 10^7$	n/a	0.29	**4079329**	**5.2**	**0.82**
unif04-62†	1296842	1.2	**1.00**	534814	**0.6**	**1.00**	3140198	3.1	0.90	**442513**	**0.6**	**1.00**
unif04-65†	$> 10^7$	n/a	0.48	1110469	1.3	**1.00**	3800951	3.7	0.84	**936079**	**1.2**	**1.00**
unif04-80†	5433833	4.7	0.68	**2016760**	**2.4**	**0.96**	$> 10^7$	n/a	0.34	2105533	2.8	0.94
unif04-83†	$> 10^7$	n/a	0.04	**5260203**	**6.3**	**0.77**	$> 10^7$	n/a	0.24	5827928	8.8	0.74
unif04-86†	$> 10^7$	n/a	0.18	**4026873**	**4.9**	**0.80**	$> 10^7$	n/a	0.49	4285016	6.0	**0.80**
unif04-91†	1826562	1.6	0.97	**538064**	**0.7**	**1.00**	2634811	2.9	0.91	572947	0.8	**1.00**
unif04-99†	$> 10^7$	n/a	0.32	4010745	**5.0**	**0.87**	$> 10^7$	n/a	0.34	**3503235**	5.2	0.81
O*1582	15032455	176.6	0.98	11250878	**159.2**	**1.00**	$> 10^8$	n/a	0.34	**10819125**	162.6	0.99
O*1583	4311571	51.0	**1.00**	**3628184**	**50.6**	**1.00**	53945903	5805.5	0.69	3714975	56.8	**1.00**
O*1584	9279077	109.2	**1.00**	8292676	115.3	**1.00**	$> 10^8$	n/a	0.40	**7139020**	**108.1**	**1.00**
O*1585	20140780	242.3	0.96	**10724723**	**155.8**	**1.00**	$> 10^8$	n/a	0.38	11512426	174.3	0.99
O*1586	19112213	**222.9**	0.94	15649195	225.5	**0.99**	$> 10^8$	n/a	0.27	**14538393**	249.3	**0.99**
O*1587	1602114	18.8	**1.00**	**1206202**	**17.9**	**1.00**	25999225	2846.3	0.96	1426090	21.3	**1.00**
O*1588	19823423	241.2	0.97	16073531	228.3	**1.00**	$> 10^8$	n/a	0.36	**14406395**	**227.0**	0.99
O*1589	7727511	90.9	**1.00**	**4813256**	**66.6**	**1.00**	95733874	10081.0	0.52	5031016	75.3	**1.00**

We compare the performance of *Hybrid* with γ=10 (the default value), *adaptG*^{2}*WSAT*$_P$, *VW*, and *R+adaptNovelty+* on the 11 groups of instances, or 65 instances, in Tables 7 and 8, in which instances with † on the right constitute the entire set of instances that were used to originally evaluate *R+adaptNovelty+* in [1]. *R+adaptNovelty+* was downloaded from `http://users.rsise.anu.edu.au/~anbu/`. From these two tables, we summarize the strengths of the performance of *Hybrid*.

1. Among the 3 algorithms *adaptG*^{2}*WSAT*$_P$, *VW*, and *R+adaptNovelty+*, *adaptG*^{2}*WSAT*$_P$ exhibits the best performance on parity, the industrial instances, and the 2 groups of random instances. *Hybrid* inherits the strengths of *adaptG*^{2}*WSAT*$_P$ on these 4 groups. Among these 3 algorithms, *VW* exhibits the best performance on SSS.1.0a and the crafted instances. *Hybrid* inherits the strengths of *VW* on these 2 groups.

2. *Hybrid* outperforms *adaptG*^{2}*WSAT*$_P$ on the following 6 groups: blocksworld, Beijing, QG, SSS.1.0a, the crafted instances, and the industrial instances. *Hybrid* outperforms *VW* on the following 8 groups: blocksworld, Beijing, GCP, parity, QG, the industrial instances, and the 2 groups of random instances. *Hybrid* outperforms *R+adaptNovelty+* on the following 7 groups: blocksworld, parity, SSS.1.0a, the crafted instances, the industrial instances, and the 2 groups of random instances.

3. Without any manual tuning parameters, *Hybrid* solves each of these 65 instances in a reasonable time. In contrast, *adaptG*^{2}*WSAT*$_P$, *VW*, and *R+adaptNovelty+* have difficulty on some of these instances.

A state-of-the-art local search algorithm can often solve a satisfiable instance quickly if this algorithm uses the optimal values of its parameters, but it is difficult to find the optimal values for every instance. Moreover, a state-of-the-art local search algorithm may be effective for one class of instances but have poor performance for another. However, as shown in Tables 7 and 8, *Hybrid* solves a broad range of instances in a reasonable time using a fixed value of γ, the default value 10. In contrast, $adaptG^2WSAT_P$, VW, and *R+adaptNovelty+* have difficulty on some of these instances. Therefore, the overall performance of *Hybrid* is much better than the overall performance of $adaptG^2WSAT_P$, VW, and *R+adaptNovelty+*, although the performance of *Hybrid* on each instance in Tables 7 and 8 is not necessarily better than the best performance of $adaptG^2WSAT_P$, VW, and *R+adaptNovelty+* on this instance.

5.4 Justification for Proposed Switching Strategy

To justify the proposed switching strategy used in *Hybrid*, we implement the other two switching strategies, namely the opposite switching strategy and the random switching strategy, in two algorithms, called *Hybrid_opposite* and *Hybrid_random*.

Table 9. Experimental results for $adaptG^2WSAT_P$, VW, *Hybrid* (γ=10) and *Hybrid_opposite* (γ=10) on structured and crafted instances. In *Hybrid* (γ=10) and *Hybrid_opposite* (γ=10), $s = 0$.

	$adaptG^3WSAT_P$			VW			*Hybrid* (γ=10)			*Hybrid_opposite* (γ=10)		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
bw_large.c†	992093	3.5	**1.00**	1868393	6.0	**1.00**	**597473**	**2.5**	0.99	1271889	5.7	**1.00**
bw_large.d	2124858	12.4	0.96	2963500	18.1	0.98	**962677**	**6.3**	0.98	1778613	13.6	**0.99**
e0ddr2*1†	3068450	15.3	0.99	6549282	22.5	0.66	**114774**	**2.9**	**1.00**	2746153	17.2	**1.00**
e0ddr2*4†	694059	4.3	**1.00**	1894243	7.9	0.98	**71214**	**2.7**	**1.00**	666626	4.8	**1.00**
enddr2*1†	641226	4.3	**1.00**	4484178	17.6	0.83	**54245**	**2.6**	**1.00**	681762	5.2	**1.00**
enddr2*8†	555475	3.8	**1.00**	3398071	16.1	0.92	**47090**	**2.7**	**1.00**	528608	4.3	**1.00**
ewddr2*1†	520705	3.6	**1.00**	4052096	16.5	0.88	**42881**	**2.5**	**1.00**	510126	4.2	**1.00**
ewddr2*8†	432671	3.1	**1.00**	4608302	20.5	0.86	**35079**	**2.4**	**1.00**	422659	3.7	**1.00**
g250.29	**637472**	**28.2**	**1.00**	$> 10^7$	n/a	0.18	1306322	94.0	0.88	$> 10^7$	n/a	0.13
logi*.c†	49469	**0.1**	**1.00**	70446	**0.1**	**1.00**	**19038**	**0.1**	**1.00**	46795	**0.1**	**1.00**
par16-1†	**55017679**	**28.0**	**1.00**	$> 10^9$	n/a	0.00	65354529	37.8	**1.00**	$> 10^9$	n/a	0.02
par16-2†	**106070896**	**57.0**	**1.00**	$> 10^9$	n/a	0.00	152549064	92.5	**1.00**	$> 10^9$	n/a	0.07
par16-3†	97156387	**51.6**	**1.00**	$> 10^9$	n/a	0.00	**87443760**	53.5	**1.00**	$> 10^9$	n/a	0.08
par16-4†	118557332	**61.4**	**1.00**	$> 10^9$	n/a	0.00	**108114087**	63.6	**1.00**	$> 10^9$	n/a	0.15
par16-5†	**83028280**	**44.4**	**1.00**	$> 10^9$	n/a	0.01	105083154	63.0	**1.00**	$> 10^9$	n/a	0.04
qg1-07†	6206	**0.0**	**1.00**	27607	0.1	**1.00**	**5722**	**0.0**	**1.00**	14097	0.1	**1.00**
qg1-08†	**380083**	**2.3**	**1.00**	2178590	30.9	0.95	440417	3.6	**1.00**	2032878	26.9	0.98
qg2-07†	4318	**0.0**	**1.00**	9847	**0.0**	**1.00**	**4226**	**0.0**	**1.00**	7259	0.1	**1.00**
qg2-08†	1540278	10.7	**1.00**	$> 10^7$	n/a	0.48	**1353338**	**10.0**	**1.00**	9215062	141.0	0.53
qg3-08†	35226	**0.1**	**1.00**	143567	0.2	**1.00**	**32808**	**0.1**	**1.00**	133873	0.3	**1.00**
qg4-09†	**61492**	**0.1**	**1.00**	318048	0.6	**1.00**	63634	**0.1**	**1.00**	150705	0.5	**1.00**
qg5-11†	20494	**0.2**	**1.00**	53713	0.5	**1.00**	**19212**	**0.2**	**1.00**	41630	1.6	**1.00**
qg6-09†	**353**	**0.0**	**1.00**	1151	**0.0**	**1.00**	417	**0.0**	**1.00**	748	0.0	**1.00**
qg7-09†	**283**	**0.0**	**1.00**	1074	**0.0**	**1.00**	320	**0.0**	**1.00**	684	0.0	**1.00**
qg7-13†	$> 10^8$	n/a	0.48	8843466	307.6	**0.90**	**1881094**	**32.3**	0.71	$> 10^8$	n/a	0.40
*bug3	$> 10^7$	n/a	0.00	1786329	3.7	**0.98**	**628668**	**3.0**	0.97	$> 10^7$	n/a	0.02
*bug4	$> 10^7$	n/a	0.03	185184	**0.4**	**1.00**	**113857**	0.8	**1.00**	$> 10^7$	n/a	0.08
*bug5	$> 10^7$	n/a	0.06	280071	**0.7**	**1.00**	**102743**	0.9	**1.00**	$> 10^7$	n/a	0.01
*bug17	128497	2.6	0.70	32999	**0.3**	**1.00**	**20361**	1.4	**1.00**	501997	11.3	0.51
*bug38	$> 10^7$	n/a	0.13	**157834**	**0.4**	**1.00**	210259	1.2	0.96	$> 10^7$	n/a	0.05
*bug39	$> 10^7$	n/a	0.39	**83287**	**0.2**	**1.00**	89306	0.7	**1.00**	$> 10^7$	n/a	0.23
*bug40	$> 10^7$	n/a	0.12	98834	**0.3**	**1.00**	**55004**	0.6	**1.00**	$> 10^7$	n/a	0.13
*bug59	$> 10^7$	n/a	0.49	66090	**0.3**	**1.00**	**30058**	1.8	**1.00**	$> 10^7$	n/a	0.49
g*1334	$> 10^8$	n/a	0.10	167786	**0.2**	**1.00**	**67540**	**0.2**	**1.00**	$> 10^8$	n/a	0.04
g*1335	18665469	17.5	0.54	170227	**0.2**	**1.00**	**76130**	0.3	**1.00**	40211337	59.3	0.52
g*1336	137026	0.3	0.62	132881	**0.2**	**1.00**	**77116**	0.3	**1.00**	263465	0.6	0.57
g*1337	94475014	100.4	0.51	304756	**0.5**	**1.00**	**142529**	0.6	**1.00**	$> 10^8$	n/a	0.46
g*1338	543830	2.2	0.56	822693	**1.9**	**1.00**	**373965**	2.0	**1.00**	6839090	14.1	0.51
g*1339	1040992	5.7	0.62	1220583	**2.9**	**1.00**	**606668**	4.4	**1.00**	1132979	8.4	0.57
g*1340	$> 10^8$	n/a	0.00	8891929	22.4	**1.00**	**2529897**	**14.0**	**1.00**	$> 10^8$	n/a	0.00
g*1341	$> 10^8$	n/a	0.00	6253863	**17.8**	**1.00**	**2751076**	23.1	**1.00**	$> 10^8$	n/a	0.00

Table 10. Experimental results for *adaptG*$^2WSAT_P$, VW, $Hybrid$ (γ=10),and $Hybrid\_opposite$ (γ=10) on industrial and random instances. In $Hybrid$ (γ=10) and $Hybrid\_opposite$ (γ=10), $s = 0$.

	adaptG$^2WSAT_P$			VW			$Hybrid$ (γ=10)			$Hybrid\_opposite$ (γ=10)		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
v*1912	**3419845**	101.6	**1.00**	61152892	3037.7	0.68	3570353	**95.6**	**1.00**	41774649	2103.1	0.77
v*1915	**11570303**	**372.6**	**1.00**	$> 10^8$	n/a	0.18	11904448	399.1	**1.00**	$> 10^8$	n/a	0.14
v*1923	**1300954**	31.1	**1.00**	12518563	428.5	0.99	1404437	**28.2**	**1.00**	8674682	367.8	**1.00**
v*1924	1746729	41.7	**1.00**	13744232	515.7	0.99	**1537351**	**35.0**	**1.00**	12814347	503.8	**1.00**
v*1944	3587804	221.8	**1.00**	58541545	7971.7	0.69	**3508563**	**194.1**	**1.00**	49909890	6017.3	0.74
v*1955	1393168	65.4	**1.00**	10396220	1074.0	**1.00**	**1336078**	**50.9**	**1.00**	10218659	691.4	**1.00**
v*1956	**1494423**	**70.0**	**1.00**	13419375	1437.0	0.98	1607320	**70.0**	**1.00**	13151370	1096.5	**1.00**
v*1959	559281	29.9	**1.00**	11433482	1377.2	**1.00**	**542837**	**26.4**	**1.00**	11135528	1103.1	**1.00**
unif04-52†	4656882	**5.2**	0.79	$> 10^7$	n/a	0.29	**4079329**	**5.2**	**0.82**	> 10	n/a	0.33
unif04-62†	534814	**0.6**	**1.00**	3140198	3.1	0.90	**442513**	**0.6**	**1.00**	8128046	10.8	0.58
unif04-65†	1110469	1.3	**1.00**	3800951	3.7	0.84	**936079**	**1.2**	**1.00**	4278880	5.3	0.72
unif04-80†	**2016760**	**2.4**	**0.96**	$> 10^7$	n/a	0.34	2105533	2.8	0.94	$> 10^7$	n/a	0.18
unif04-83†	**5260203**	**6.3**	**0.77**	$> 10^7$	n/a	0.24	5827928	8.8	0.74	$> 10^7$	n/a	0.20
unif04-86†	**4026873**	**4.9**	**0.80**	$> 10^7$	n/a	0.50	4285016	6.0	**0.80**	$> 10^7$	n/a	0.42
unif04-91†	**538064**	**0.7**	**1.00**	2634811	2.9	0.91	572947	0.8	**1.00**	6003885	9.2	0.69
unif04-99†	4010745	**5.0**	**0.87**	$> 10^7$	n/a	0.34	**3503235**	5.2	0.81	$> 10^7$	n/a	0.14
O*1582	11250878	**159.2**	**1.00**	$> 10^8$	n/a	0.34	**10819125**	162.6	0.99	$> 10^8$	n/a	0.10
O*1583	**3628184**	**50.6**	**1.00**	53945903	5805.5	0.69	3714975	56.8	**1.00**	$> 10^8$	n/a	0.18
O*1584	8292676	115.3	**1.00**	$> 10^8$	n/a	0.40	**7139020**	**108.1**	**1.00**	$> 10^8$	n/a	0.16
O*1585	**10724723**	**155.8**	**1.00**	$> 10^8$	n/a	0.38	11512426	174.3	0.99	$> 10^8$	n/a	0.11
O*1586	15649195	**225.5**	**0.99**	$> 10^8$	n/a	0.27	**14538393**	249.3	**0.99**	$> 10^8$	n/a	0.07
O*1587	**1206202**	17.9	**1.00**	25999225	2846.3	0.96	1426090	21.3	**1.00**	$> 10^8$	n/a	0.42
O*1588	16073531	228.3	**1.00**	$> 10^8$	n/a	0.36	**14406395**	**227.0**	0.99	$> 10^8$	n/a	0.09
O*1589	**4813256**	**66.6**	**1.00**	95733874	10081.0	0.52	5031016	75.3	**1.00**	$> 10^8$	n/a	0.12

We compare the switching strategies used in $Hybrid$, $Hybrid\_opposite$, and $Hybrid\_random$. We first recall the switching strategy used in $Hybrid$. In each 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. $Hybrid\_opposite$ uses the opposite switching strategy to that used in $Hybrid$. In each step, $Hybrid\_opposite$ chooses a variable to flip according to *heuristic* VW if the distribution of variable weights is even, and selects a variable to flip according to *heuristic adaptG*$^2WSAT_P$ otherwise. $Hybrid\_random$ uses the random switching strategy. In each step, $Hybrid\_random$ chooses a variable to flip according to *heuristic* VW or *heuristic adaptG*$^2WSAT_P$. $Hybrid\_random$ selects a heuristic from *heuristic* VW and *heuristic adaptG*$^2WSAT_P$ randomly, not based on the distribution of variable weights.

We compare the performance of $Hybrid$ with that of $Hybrid\_opposite$ on the 11 groups of instances, or 65 instances, in Tables 9 and 10. The value of parameter γ in both $Hybrid$ and $Hybrid\_opposite$ is set to 10. Among these 65 in stances, $Hybrid\_opposite$ does not show better performance than $Hybrid$ on any instance. In fact, $Hybrid\_opposite$ inherits all of the weaknesses of *adaptG*$^2WSAT_P$ and VW. Specifically, $Hybrid\_opposite$ inherits the poor performance of *adaptG*$^2WSAT_P$ on qg7-13, the 8 instances in SSS.1.0a, and the 8 crafted instances, and inherits the poor performance of VW on g250.29, the 5 instances in parity, the 8 industrial instances, and the 8 random instances from the SAT 2005 competition benchmark (instances from O*1582 to O*1589).

We compare the performance of $Hybrid$ with that of $Hybrid\_random$ on the 11 groups of instances, or 65 instances, in Tables 11 and 12. The value of parameter γ in $Hybrid$ is set to 10. The run time performance of $Hybrid\_random$ is better than that of $Hybrid$ on only 11 out of the 65 instances presented in Tables 11 and 12. On the remaining 54 instances, $Hybrid$ shows better run time performance than $Hybrid\_random$. Specifically, the run time performance of $Hybrid$ is much better than that of $Hybrid\_random$ on GCP, qg7-13, v*1915, and the 8 random instances from the SAT 2005 competition benchmark.

Table 11. Experimental results for *adaptG*$^2WSAT_P$, *VW*, *Hybrid* (γ=10),and *Hybrid_random* on the structured and crafted instances. In *Hybrid* (γ=10) and *Hybrid_random*, $s = 0$.

	$adaptG^3WSAT_P$			VW			$Hybrid$ (γ=10)			$Hybrid\_random$		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
bw_large.c†	992093	3.5	**1.00**	1868393	6.0	**1.00**	**597473**	**2.5**	0.99	**376008**	**1.6**	**1.00**
bw_large.d	2124858	12.4	0.96	2963500	18.1	0.98	**962677**	**6.3**	0.98	**397443**	**3.0**	**1.00**
e0ddr2*1†	3068450	15.3	0.99	6549282	22.5	0.66	**114774**	**2.9**	**1.00**	161557	3.0	**1.00**
e0ddr2*4†	694059	4.3	**1.00**	1894243	7.9	0.98	**71214**	**2.7**	**1.00**	165399	3.2	**1.00**
enddr2*1†	641226	4.3	**1.00**	4484178	17.6	0.83	**54245**	**2.6**	**1.00**	116633	2.8	**1.00**
enddr2*8†	555475	3.8	**1.00**	3398071	16.1	0.92	**47090**	2.7	**1.00**	89587	**2.6**	**1.00**
ewddr2*1†	520705	3.6	**1.00**	4052096	16.5	0.88	**42881**	**2.5**	**1.00**	99759	2.8	**1.00**
ewddr2*8†	432671	3.1	**1.00**	4608302	20.5	0.86	**35079**	2.4	**1.00**	79277	**2.1**	**1.00**
g250.29	**637472**	**28.2**	**1.00**	$> 10^7$	n/a	0.18	1306322	94.0	0.88	7963853	482.3	0.59
logi*.c†	49469	**0.1**	**1.00**	70446	**0.1**	**1.00**	**19038**	**0.1**	**1.00**	**13543**	**0.1**	**1.00**
par16-1†	**55017679**	**28.0**	**1.00**	$> 10^9$	n/a	0.00	65354529	37.8	**1.00**	360851332	299.6	0.84
par16-2†	**106070896**	**57.0**	**1.00**	$> 10^9$	n/a	0.00	152549064	92.5	**1.00**	118912415	119.2	**1.00**
par16-3†	97156387	**51.6**	**1.00**	$> 10^9$	n/a	0.00	**87443760**	53.5	**1.00**	178938233	113.5	0.99
par16-4†	118557332	**61.4**	**1.00**	$> 10^9$	n/a	0.00	**108114087**	63.6	**1.00**	**56660973**	**54.9**	**1.00**
par16-5†	**83028280**	**44.4**	**1.00**	$> 10^9$	n/a	0.01	105083154	63.0	**1.00**	111896481	112.5	**1.00**
qg1-07†	6206	**0.0**	**1.00**	27607	0.1	**1.00**	**5722**	**0.0**	**1.00**	8494	**0.0**	**1.00**
qg1-08†	**380083**	**2.3**	**1.00**	2178590	30.9	0.95	440417	3.6	**1.00**	728452	5.0	**1.00**
qg2-07†	4318	**0.0**	**1.00**	9847	**0.0**	**1.00**	4226	**0.0**	**1.00**	**3895**	**0.0**	**1.00**
qg2-08†	1540278	10.7	**1.00**	$> 10^7$	n/a	0.48	**1353338**	**10.0**	**1.00**	2537061	19.5	0.92
qg3-08†	35226	**0.1**	**1.00**	143567	0.2	**1.00**	**32808**	**0.1**	**1.00**	81866	0.2	**1.00**
qg4-09†	**61492**	**0.1**	**1.00**	318048	0.6	**1.00**	63634	**0.1**	**1.00**	133997	0.3	**1.00**
qg5-11†	20494	**0.2**	**1.00**	53713	0.5	**1.00**	**19212**	**0.2**	**1.00**	16197	0.3	0.99
qg6-09†	**353**	**0.0**	**1.00**	1151	**0.0**	**1.00**	417	**0.0**	**1.00**	443	**0.0**	**1.00**
qg7-09†	**283**	**0.0**	**1.00**	1074	**0.0**	**1.00**	320	**0.0**	**1.00**	364	**0.0**	**1.00**
qg7-13†	$> 10^8$	n/a	0.48	8843466	307.6	**0.90**	**1881094**	**32.3**	0.71	7501896	156.5	0.52
*bug3	$> 10^7$	n/a	0.00	1786329	3.7	0.98	**628668**	**3.0**	0.97	1176148	6.8	**1.00**
*bug4	$> 10^7$	n/a	0.03	185184	**0.4**	**1.00**	**113857**	0.8	**1.00**	1716795	9.8	0.98
*bug5	$> 10^7$	n/a	0.06	280071	0.7	**1.00**	102743	0.9	**1.00**	**38993**	**0.3**	**1.00**
*bug17	128497	2.6	0.70	32999	**0.3**	**1.00**	20361	1.4	**1.00**	**14477**	0.6	**1.00**
*bug38	$> 10^7$	n/a	0.13	157834	0.4	**1.00**	210259	1.2	0.96	**22611**	**0.3**	**1.00**
*bug39	$> 10^7$	n/a	0.39	83287	**0.2**	**1.00**	89306	0.7	**1.00**	**19765**	**0.2**	**1.00**
*bug40	$> 10^7$	n/a	0.12	98834	**0.3**	**1.00**	55004	0.6	**1.00**	**22070**	**0.3**	**1.00**
*bug59	$> 10^7$	n/a	0.49	66090	**0.3**	**1.00**	30058	1.8	**1.00**	**19805**	0.6	**1.00**
g*1334	$> 10^8$	n/a	0.10	167786	**0.2**	**1.00**	**67540**	**0.2**	**1.00**	108151	0.3	**1.00**
g*1335	18665469	17.5	0.54	170227	**0.2**	**1.00**	**76130**	0.3	**1.00**	116019	0.3	**1.00**
g*1336	137026	0.3	0.62	132881	**0.2**	**1.00**	**77116**	0.3	**1.00**	99677	0.4	**1.00**
g*1337	94475014	100.4	0.51	304756	**0.5**	**1.00**	**142529**	0.6	**1.00**	224652	0.8	**1.00**
g*1338	543830	2.2	0.56	822693	**1.9**	**1.00**	**373965**	2.0	**1.00**	679395	3.2	**1.00**
g*1339	1040992	5.7	0.62	1220583	**2.9**	**1.00**	**606668**	4.4	**1.00**	1042199	6.0	**1.00**
g*1340	$> 10^8$	n/a	0.00	8891929	22.4	**1.00**	**2529897**	**14.0**	**1.00**	10063627	44.9	**1.00**
g*1341	$> 10^8$	n/a	0.00	6253863	**17.8**	**1.00**	**2751076**	23.1	**1.00**	8522296	46.2	**1.00**

Table 12. Experimental results for *adaptG*$^2WSAT_P$, *VW*, *Hybrid* (γ=10),and *Hybrid_random* on the industrial and random instances. In *Hybrid* (γ=10) and *Hybrid_random*, $s = 0$.

	$adaptG^2WSAT_P$			VW			$Hybrid$ (γ=10)			$Hybrid\_random$		
	#flips	time	suc	#flips	time	suc	#flips	time	suc	#flips	time	suc
v*1912	**3419845**	101.6	**1.00**	61152892	3037.7	0.68	3570353	**95.6**	**1.00**	8857485	263.6	**1.00**
v*1915	**11570303**	**372.6**	**1.00**	$> 10^8$	n/a	0.18	11904448	399.1	**1.00**	37407782	1426.5	0.84
v*1923	**1300954**	31.1	**1.00**	12518563	428.5	0.99	1404437	**28.2**	**1.00**	2218590	47.9	**1.00**
v*1924	1746729	41.7	**1.00**	13744232	515.7	0.99	**1537351**	**35.0**	**1.00**	2920894	79.4	**1.00**
v*1944	3587804	221.8	**1.00**	58541545	7971.7	0.69	**3508563**	**194.1**	**1.00**	7194486	430.0	**1.00**
v*1955	1393168	65.4	**1.00**	10396220	1074.0	**1.00**	**1336078**	**50.9**	**1.00**	2224233	87.1	**1.00**
v*1956	**1494423**	**70.0**	**1.00**	13419375	1437.0	0.98	1607320	**70.0**	**1.00**	2382822	112.4	**1.00**
v*1959	559281	29.9	**1.00**	11433482	1377.2	**1.00**	**542837**	**26.4**	**1.00**	1465191	76.5	**1.00**
unif04-52†	4656882	**5.2**	0.79	$> 10^7$	n/a	0.29	**4079329**	**5.2**	**0.82**	6152594	7.1	0.67
unif04-62†	534814	**0.6**	**1.00**	3140198	3.1	0.90	**442513**	**0.6**	**1.00**	1706443	2.1	0.94
unif04-65†	1110469	1.3	**1.00**	3800951	3.7	0.84	**936079**	**1.2**	**1.00**	1973233	2.2	0.94
unif04-80†	**2016760**	**2.4**	**0.96**	$> 10^7$	n/a	0.34	2105533	2.8	0.94	$> 10^7$	n/a	0.33
unif04-83†	**5260203**	**6.3**	**0.77**	$> 10^7$	n/a	0.24	5827928	8.8	0.74	7021946	8.8	0.60
unif04-86†	**4026873**	**4.9**	**0.80**	$> 10^7$	n/a	0.50	4285016	6.0	**0.80**	$> 10^7$	n/a	0.49
unif04-91†	**538064**	**0.7**	**1.00**	2634811	2.9	0.91	572947	0.8	**1.00**	1224722	1.6	0.98
unif04-99†	4010745	**5.0**	**0.87**	$> 10^7$	n/a	0.34	**3503235**	5.2	0.81	$> 10^7$	n/a	0.46
O*1582	11250878	**159.2**	**1.00**	$> 10^8$	n/a	0.34	**10819125**	162.6	0.99	26539748	517.4	0.94
O*1583	**3628184**	**50.6**	**1.00**	53945903	5805.5	0.69	3714975	56.8	**1.00**	7616784	149.5	**1.00**
O*1584	8292676	115.3	**1.00**	$> 10^8$	n/a	0.40	**7139020**	**108.1**	**1.00**	16892628	325.9	0.99
O*1585	**10724723**	**155.8**	**1.00**	$> 10^8$	n/a	0.38	11512426	174.3	0.99	28287602	560.2	0.93
O*1586	15649195	**225.5**	**0.99**	$> 10^8$	n/a	0.27	**14538393**	249.3	**0.99**	37189904	708.2	0.84
O*1587	**1206202**	17.9	**1.00**	25999225	2846.3	0.96	1426090	21.3	**1.00**	2979337	56.8	**1.00**
O*1588	16073531	228.3	**1.00**	$> 10^8$	n/a	0.36	**14406395**	**227.0**	0.99	29788687	572.4	0.89
O*1589	**4813256**	**66.6**	**1.00**	95733874	10081.0	0.52	5031016	75.3	**1.00**	10937851	212.4	**1.00**

6. Conclusion

We have proposed a new switching criterion: the evenness or unevenness of the distribution of variable weights. Then, to evaluate the effectiveness of this criterion, we have developed a new local search algorithm *Hybrid*, which switches between *heuristic* $adaptG^2WSAT_P$ and *heuristic VW* in every step according to this switching criterion. This new algorithm combines intensification and diversification by switching between these two heuristics. Our experimental results show that the strengths of the algorithms $adaptG^2WSAT_P$ and VW are combined in the single algorithm $Hybrid$.

Acknowledgements

The authors wish to thank two anonymous referees for their thoughtful and insightful comments and suggestions that helped improve the paper.

References

[1] Anbulagan, D. N. Pham, J. Slaney, and A. Sattar. Old Resolution Meets Modern SLS. In *Proceedings of AAAI-2005* (2005), 354–359.

[2] I. P. Gent and T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. In *Proceedings of AAAI-1993* (1993), 28–33.

[3] E. A. Hirsch and A. Kojevnikov. UnitWalk: A New SAT Solver that Uses Local Search Guided by Unit Clause Elimination. *Annals of Mathematics and Artificial Intelligence*, **43**(1) (2005), 91–111.

[4] H. H. Hoos. An Adaptive Noise Mechanism for WalkSAT. In *Proceedings of AAAI-2002* (2002), 655–660.

[5] H. H. Hoos and T. Stűtzle. *Stochastic Local Search: Foundations and Applications.* Morgan Kaufmann (2004).

[6] F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and Probabilistic Smoothing: Efficient Dynamical Local Search for SAT. In *Proceedings of CP-2002*, Springer, *Lecture Notes in Comput. Sci.* **2470** (2002), 233–248.

[7] C. M. Li and W. Q. Huang. Diversification and Determinism in Local Search for Satisfiability. In *Proceedings of SAT-2005*, Springer, *Lecture Notes in Comput. Sci.* **3569** (2005), 158–172.

[8] C. M. Li, W. Wei, and H. Zhang. Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In *Proceedings of LSCS-2006* (2006), 2–16.

[9] C. M. Li, W. Wei, and H. Zhang. Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In Frédéric Benhamou, Narendra Jussien, and Barry O'Sullivan, editors, *Trends in Constraint Programming.* ISTE (2007), chapter 14, 261–267.

[10] C. M. Li, W. Wei, and H. Zhang. Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In *Proceedings of SAT-2007*, Springer, *Lecture Notes in Comput. Sci.* **4501** (2007), 121–133.

[11] X. Y. Li, M. F. Stallmann, and F. Brglez. A Local Search SAT Solver Using an Effective Switching Strategy and an Efficient Unit Propagation. In *Proceedings of SAT-2003*, Springer, *Lecture Notes in Comput. Sci.* **2919** (2003), 53–68.

[12] D. A. McAllester, B. Selman, and H. Kautz. Evidence for Invariant in Local Search. In *Proceedings of AAAI-1997* (1997), 321–326.

[13] P. Mills and E. Tsang. Guided Local Search for Solving SAT and Weighted MAXSAT Problems. *Journal of Automated Reasoning, Special Issue on Satisfiability Problems*, **24** (2000), 205–223.

[14] P. Morris. The Breakout Method for Escaping from Local Minima. In *Proceedings of AAAI-1993* (1993), 40–45.

[15] S. Prestwich. Random Walk with Continuously Smoothed Variable Weights. In *Proceedings of SAT-2005*, Springer, *Lecture Notes in Comput. Sci.* **3569** (2005), 203–215.

[16] D. Schuurmans and F. Southey. Local Search Characteristics of Incomplete SAT Procedures. In *Proceedings of AAAI-2000* (2000), 297–302.

[17] B. Selman, H. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proceedings of AAAI-1994* (1994), 337–343.

[18] B. Selman, D. Mitchell, and H. Levesque. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of AAAI-1992* (1992), 440–446.

[19] J. Thornton, D. N. Pham, S. Bain, and V. F. Jr. Additive versus Multiplicative Clause Weighting for SAT. In *Proceedings of AAAI-2004* (2004), 191–196.

[20] D. A. D. Tompkins and H. H. Hoos. Warped Landscapes and Random Acts of SAT Solving. AI&M 26-2004. In *Proceedings of AI&M-2004* (2004).

[21] W. Wei, C. M. Li, and H. Zhang. Criterion for Intensification and Diversification in Local Search for SAT. In *Proceedings of LSCS-2007* (2007), 16–30.

[22] Z. Wu and B. W. Wah. Global-Search Strategy in Discrete Lagrangian Methods for Solving Hard Satiafiability Problems. In *Proceedings of AAAI-2000* (2000), 310–315.