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

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

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

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

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 adaptG²WSATp and VW select a variable to flip, as heuristic adaptG²WSATp and heuristic VW, respectively. To evaluate the effectiveness of this criterion, we apply it to heuristic adaptG²WSATp 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 adaptG²WSATp 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 adaptG²WSATp and VW, and exhibits generally better performance than adaptG²WSATp 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 adaptG²WSATp, 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], G²WSAT [7], and adaptG²WSATP [8, 9]. Among these algorithms, R+adaptNovelty+ integrates restricted resolution in a preprocessing phase into AdaptNovelty+ [4], G²WSAT deterministically uses promising decreasing variables, and adaptG²WSATP implements the adaptive noise mechanism from [4] in G²WSAT 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+, G²WSAT 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. Experiments in [8, 9] show that, without any manual noise or other parameter tuning, adaptG²WSATP shows generally good performance, compared with G²WSAT with optimal static noise settings, or is sometimes even better than G²WSAT, and that adaptG²WSATP 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 adaptG²WSATP, and the algorithm from the second class is VW. Our experimental results show that the performance of adaptG²WSATP 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 adaptG²WSATP 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. G²WSAT [7] switches between a variant of GSAT and Novelty++. The local search algorithm adaptG²WSATP [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 \( \text{adapt}G^2W\text{SAT}_P \) and \( \text{VW} \) select a variable to flip, as heuristic \( \text{adapt}G^2W\text{SAT}_P \) and heuristic \( \text{VW} \), respectively. Then, to evaluate the effectiveness of this switching criterion, we develop a new local search algorithm called \( \text{Hybrid} \), which switches between heuristic \( \text{adapt}G^2W\text{SAT}_P \) and heuristic \( \text{VW} \) in every step according to this switching criterion. This new algorithm allows suitable diversification strategies to complement intensification strategies by switching between heuristic \( \text{adapt}G^2W\text{SAT}_P \) and heuristic \( \text{VW} \). Our experimental results show that, on a broad range of SAT instances presented in this paper, \( \text{Hybrid} \) inherits the strengths of \( \text{adapt}G^2W\text{SAT}_P \) and \( \text{VW} \).

2. Review of Algorithms \( \text{adapt}G^2W\text{SAT}_P \) and \( \text{VW} \)

Given a CNF formula \( F \) and an assignment \( A \), the objective function that local search for SAT attempts to minimize is usually the total number of unsatisfied clauses in \( F \) under \( A \). Let \( x \) be a variable. The break of \( x \), \( \text{break}(x) \), is the number of clauses in \( F \) that are currently satisfied but will be unsatisfied if \( x \) is flipped. The make of \( x \), \( \text{make}(x) \), is the number of clauses in \( F \) that are currently unsatisfied but will be satisfied if \( x \) is flipped. The score of \( x \) with respect to \( A \), \( \text{score}_A(x) \), is the difference between \( \text{make}(x) \) and \( \text{break}(x) \). Let \( \text{best} \) and \( \text{second} \) be the best and second best variables in a randomly selected unsatisfied clause \( c \) according to their scores. Heuristic \( \text{Novelty} \) [12] selects a variable to flip from \( c \) as follows.

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

Given a CNF formula \( F \) and an assignment \( A \), a variable \( x \) is said to be decreasing with respect to \( A \) if \( \text{score}_A(x) > 0 \). 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 \( \text{score}_C(x) > 0 \) 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 \( A \), \( \text{pscore}_A(x) \), is defined in [8, 9, 10] as

\[ \text{pscore}_A(x) = \text{score}_A(x) + \text{score}_B(x') \]
where $\text{score}_A(x)$ is the score of $x$ with respect to $A$ and $\text{score}_B(x')$ is the score of $x'$ with respect to $B$.\footnote{2}

If there are promising decreasing variables with respect to $B$, $\text{p-score}_A(x)$ represents the improvement in the number of unsatisfied clauses under $A$ by flipping $x$ and then $x'$. In this case, $\text{p-score}_A(x) > \text{score}_A(x)$. If there is no promising decreasing variable with respect to $B$,

$$\text{p-score}_A(x) = \text{score}_A(x).$$

Heuristic Novelty++ \footnote{8, 9} selects a variable to flip from $c$ as follows.

**Novelty++** $(p, dp)$: 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 $\text{p-score(}\text{second}) \geq \text{p-score(best)}$.

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

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

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

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, \footnote{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,
Switching Criterion in Local Search for SAT

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. 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 and includes the 8 random instances from O*1582 to O*1589. The second class is from Miroslav Velev’s SAT Benchmarks and consists of all of the formulas from Superscalar Suite 1.0a (SSS.1.0a) except for *bug54. 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 | 1.000 | 0.25 | 9.73  | 0.005 | 1.015 | 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 γ > 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\(^\ast\) 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.
Switching Criterion in Local Search for SAT

Algorithm: Hybrid(SAT-formula $F$)

1: $A$← randomly generated truth assignment;
2: for each variable $x$ do initialize $\text{flip\_time}[x]$ and $\text{var\_weight}[x]$ to 0;
3: initialize $p$, $dp$, $max\_weight$, and $ave\_weight$ to 0;
4: store promising decreasing variables in stack $\text{DecVar}$;
5: for $\text{flip} ← 1$ to $\text{Maxsteps}$ do
6: if $A$ satisfies $F$ then return $A$;
7: if $\text{max\_weight} \geq \gamma \times \text{ave\_weight}$ then
8: $y$← heuristic $\text{VW}(p)$;
9: else $y$← heuristic $\text{adaptG}^2\text{WSAT}_P(p, dp)$;
10: $A$← $A$ with $y$ flipped; adapt $p$ and $dp$;
11: update $\text{flip\_time}[y]$, $\text{var\_weight}[y]$, $\text{max\_weight}$, $\text{ave\_weight}$, and $\text{DecVar}$;
12: return Solution not found;

Figure 1. Algorithm Hybrid

weights is uneven, and selects a variable to flip according to heuristic $\text{adaptG}^2\text{WSAT}_P$ 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 $\text{adaptG}^2\text{WSAT}_P$, $\text{VW}$, and $R+\text{adaptNovelty}$ 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\textsuperscript{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,\textsuperscript{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.\textsuperscript{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 $R$+adapt$Novelty+$ \cite{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.

\textsuperscript{9.} http://www.satlib.org/
\textsuperscript{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.
\textsuperscript{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 \( s = 0 \), 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 \( \gamma \) 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 \( \gamma = 4, 10, 15, 20, 25, 30, 35, 40, \) and 45. Our experimental results show that on the hardest instances from the 11 groups, Hybrid with \( \gamma = 10 \) exhibits the best overall performance among all of these versions. So, in Hybrid, the default value of \( \gamma \) is set to 10.

### Table 3. Experimental results for adapt\( G^2WSAT_P \), VW, Hybrid\_A, Hybrid (\( \gamma = 10 \)), and Hybrid\_45 on the hardest instances from the first category. In Hybrid\_A, Hybrid (\( \gamma = 10 \)), and Hybrid\_45, \( s = 0 \).

|            | adapt\_A | Hybrid\_A | Hybrid | Hybrid\_45 |
|------------|----------|-----------|--------|------------|
| \( g250.29 \) | 99.98%   | 59.17%    | 21.25% | 50.09      |
| \#flips    | 637472   | > 10^7    | > 10^7 | 1306322    |
| time       | 28.2     | n/a       | n/a    | 94.0       |
| suc        | 1.00     | 0.18      | 0.00   | 0.88       |
| r\_unev    | 21.86%   | 1.00%     | 0.00%  |            |
|            |          | n/a       | n/a    |            |
| \( par16-2 \) |          |           |        |            |
| \#flips    | 106070896| > 10^9    | > 10^9 | 867375405  |
| time       | 57.0     | n/a       | n/a    | 540.5      |
| suc        | 1.00     | 0.00      | 0.54   | 1.00       |
| r\_unev    | 21.86%   | 1.00%     | 0.00%  |            |
|            |          | n/a       | n/a    |            |
| \( v*1915 \) |          |           |        |            |
| \#flips    | 11570303 | > 10^9    | > 10^9 | 101655555  |
| time       | 372.6    | n/a       | n/a    | 416.5      |
| suc        | 1.00     | 0.18      | 1.00   | 1.00       |
| r\_unev    | 21.86%   | 1.00%     | 0.00%  |            |
|            |          | n/a       | n/a    |            |
| \( unif04-83 \) |         |           |        |            |
| \#flips    | 5260203  | > 10^7    | > 10^7 | 5856586    |
| time       | 6.3      | n/a       | n/a    | 7.9        |
| suc        | 0.77     | 0.24      | 0.66   | 0.74       |
| r\_unev    | 21.86%   | 1.00%     | 0.00%  |            |
|            |          | n/a       | n/a    |            |
| \( O*1586 \) |          |           |        |            |
| \#flips    | 15649195 | > 10^9    | > 10^9 | 15169011   |
| time       | 225.5    | n/a       | n/a    | 233.8      |
| suc        | 0.99     | 0.27      | 0.98   | 0.99       |
| r\_unev    | 21.86%   | 1.00%     | 0.00%  |            |

Tables 3, 4, and 5 compare the performance of Hybrid (\( \gamma = 10 \)), Hybrid\_A (Hybrid with \( \gamma = 4 \)), and Hybrid\_45 (Hybrid with \( \gamma = 45 \)) 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 adapt\( G^2WSAT_P \), those that are not hard for the algorithm VW but are hard for the algorithm adapt\( G^2WSAT_P \), 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 (\( \gamma = 10 \)) is generally lower than 50%.

As a result, in most steps, Hybrid (\( \gamma = 10 \)) usually chooses heuristic adapt\( G^2WSAT_P \) to

\(^{12}\) In these three tables, adapt\_A and Hybrid refer to adapt\( G^2WSAT_P \) and Hybrid (\( \gamma = 10 \)), respectively. Results in italics indicate the poorest performance for an instance.
Table 4. Experimental results for adapt$G^2W\text{SAT}_p$, VW, Hybrid$_A$, Hybrid ($\gamma=10$), and Hybrid$_{45}$ on the hardest instances from the second category. In Hybrid$_A$, Hybrid ($\gamma=10$), and Hybrid$_{45}$, $s=0$.

|       | adapt$*$ | VW   | Hybrid$_A$ | Hybrid | Hybrid$_{45}$ |
|-------|----------|------|------------|--------|---------------|
|       | $r_{unev}$ | #flips | time (s)  | suc    | $r_{unev}$ | #flips | time (s)  | suc    | $r_{unev}$ | #flips | time (s)  | suc |
| qg7-13| 0.48      | $>10^8$ | 307.6      | 0.76   | 32.3        | $>10^8$ | 0.71      | 0.44   |
|       |           | 8843466 | 2581390    | 1881094 | 42.53%      |       | 151.5     | 0.71   |
|       |           |       |           | 0.76   |            |       |           | 0.71   |
| *bug3 | 0.00      | $>10^7$ | 1786329    | 3.7    | 3.0         | $>10^7$ | 3.1       | 0.97   |
|       |           |       |           | 3.7    |            |       |           | 0.97   |
| g*1341| 0.00      | $>10^8$ | 6253863    | 17.8   | 1.00        | $>10^8$ | 20.1      | 1.00   |
|       |           |       |           | 17.8   |            |       |           | 1.00   |

select a variable to flip. Conversely, for g250.29, $r_{unev}$ in Hybrid$_A$ is too high, as high as 99.98%, resulting in the poor performance of Hybrid$_A$ on this instance. For the second category, which consists of qg7-13, *bug3, and g*1341, $r_{unev}$ in Hybrid ($\gamma=10$) is generally higher than 50%. Consequently, in most steps, Hybrid ($\gamma=10$) 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 ($\gamma=10$) 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 adapt$G^2W\text{SAT}_p$, VW, Hybrid$_A$, Hybrid ($\gamma=10$), and Hybrid$_{45}$ on the hardest instances from the third category. In Hybrid$_A$, Hybrid ($\gamma=10$), and Hybrid$_{45}$, $s=0$.

|       | adapt$*$ | VW   | Hybrid$_A$ | Hybrid | Hybrid$_{45}$ |
|-------|----------|------|------------|--------|---------------|
|       | $r_{unev}$ | #flips | time (s)  | suc    | $r_{unev}$ | #flips | time (s)  | suc    | $r_{unev}$ | #flips | time (s)  | suc |
| bw_large.d | 2124858 | 2963500 | 12.4      | 1.00   | 99.89%       | 2683350 | 17.8      | 1.00   |
|       |          |       |           | 1.00   |            |       |           | 1.00   |
| e0ddr2*1 | 3068450 | 6549282 | 15.3      | 0.99   | 100%        | 105122 | 22.5      | 0.99   |
|       |          |       |           | 0.99   |            |       |           | 0.99   |
| logi*.c | 49469   | 70446 | 0.1       | 1.00   | 99.21%       | 17602 | 0.1       | 1.00   |
|       |          |       |           | 0.1    |            |       |           | 0.1    |

We allow Hybrid to adjust $s$ in the same way as does VW ($s>0$), and we call this version of Hybrid $H_{as}$VW. Our experimental results show that Hybrid ($\gamma=10$) exhibits better overall performance than $H_{as}$VW ($\gamma=10$) 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 $H_{asWV}$ ($\gamma=10$) and $Hybrid$ ($\gamma=10$) on the hardest instances from the 11 groups. In $H_{asWV}$ ($\gamma=10$), $s > 0$, while in $Hybrid$ ($\gamma=10$), $s = 0$.

| $H_{asWV}$ | $Hybrid$ |
|------------|-----------|
| #flips | time | suc | #flips | time | suc |
| $bw$ | 1881240 | 15.9 | 0.98 | 962077 | 6.3 | 0.98 |
| $ebddr^2*1$ | 240223 | 4.0 | 1.00 | 114774 | 2.9 | 1.00 |
| $g250.29$ | 689661 | 39.0 | 1.00 | 1306322 | 94.0 | 0.88 |
| $logi^4.c$ | 37765 | 0.1 | 1.00 | 19038 | 0.1 | 1.00 |
| $par16-2$ | 99909500 | 60.8 | 1.00 | 152549064 | 92.5 | 1.00 |
| $qg7-13$ | $>10^6$ | n/a | 0.44 | 1881094 | 32.3 | 0.71 |
| $bug5$ | $>10^6$ | 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 | 391.1 | 1.00 |
| $mi**04-83$ | 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 $R+adaptNovelty+$, $adaptG^2W SAT_P$, $VW$, and $Hybrid$ ($\gamma=10$) on the structured and crafted instances. In $Hybrid$ ($\gamma=10$), $s = 0$.

| $R+adaptNovelty+$ | $adaptG^2W SAT_P$ | $VW$ | $Hybrid$ ($\gamma=10$) |
|-------------------|-------------------|-----|-----------------------|
| #flips | time | suc | #flips | time | suc | #flips | time | suc |
| $bw$ | 9499817 | 29.1 | 0.92 | 9920993 | 3.5 | 1.00 | 1868393 | 6.0 | 1.00 | 597473 | 2.5 | 0.99 |
| $par16-31$ | 80339283 | 37.6 | 1.00 | 56017679 | 28.0 | 1.00 | $>10^9$ | n/a | 0.00 | 65354239 | 37.8 | 1.00 |
| $par16-2$ | 324926713 | 157.5 | 0.49 | 106070886 | 57.0 | 1.00 | $>10^9$ | n/a | 0.00 | 152549064 | 92.5 | 1.00 |
| $par16-31$ | 224148056 | 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 | 11857332 | 61.4 | 1.00 | $>10^9$ | n/a | 0.00 | 108114087 | 63.6 | 1.00 |
| $par16-5$ | 264879791 | 125.9 | 0.94 | 83028260 | 44.4 | 1.00 | $>10^9$ | n/a | 0.00 | 10593154 | 63.9 | 1.00 |
| $g250.29$ | 734246 | 23.2 | 1.00 | 637472 | 28.2 | 1.00 | $>10^9$ | n/a | 0.18 | 1306322 | 94.0 | 0.88 |
| $logi^4.c$ | 57693 | 0.1 | 1.00 | 94699 | 0.1 | 1.00 | 7046 | 0.1 | 1.00 | 19038 | 0.1 | 1.00 |
| $par16-31$ | 80339283 | 37.6 | 1.00 | 56017679 | 28.0 | 1.00 | $>10^9$ | n/a | 0.00 | 65354239 | 37.8 | 1.00 |
| $g250.29$ | 734246 | 23.2 | 1.00 | 637472 | 28.2 | 1.00 | $>10^9$ | n/a | 0.00 | 87443760 | 53.5 | 1.00 |
| $par16-4$ | 274054172 | 129.7 | 0.92 | 11857332 | 61.4 | 1.00 | $>10^9$ | n/a | 0.00 | 108114087 | 63.6 | 1.00 |
| $par16-5$ | 264879791 | 125.9 | 0.94 | 83028260 | 44.4 | 1.00 | $>10^9$ | n/a | 0.00 | 10593154 | 63.9 | 1.00 |

**Switching Criterion in Local Search for SAT**

- $\gamma = 10,$ $s > 0$, while in $Hybrid$ ($\gamma=10$), $s = 0$. 
- $R+adaptNovelty+$, $adaptG^2W SAT_P$, $VW$, and $Hybrid$ ($\gamma=10$) on the structured and crafted instances.
5.3 Comparison of Performance of Hybrid with Performance of adaptG^2WSAT_P, VW, and R+adaptNovelty+

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

|                  | R+adaptNovelty+ | adaptG^2WSAT_P | VW                  | Hybrid (γ=10) |
|------------------|----------------|---------------|-------------------|--------------|
|                   | #flips | time | suc | #flips | time | suc | #flips | time | suc | #flips | time | suc | #flips | time | suc |
| v*1912            | 6812718 | 148.7 | 1.00 | 34198455 | 101.6 | 1.00 | 3112592 | 3037.7 | 0.68 | 3570353 | 95.6 | 1.00 |
| v*1915            | 7890967 | 2908.9 | 0.59 | 15703038 | 372.6 | 1.00 | 11944448 | 399.1 | 1.00 | 10724723 | 28.2 | 1.00 |
| v*1923            | 2738569 | 51.7 | 1.00 | 1300954 | 31.1 | 1.00 | 12518563 | 428.5 | 0.99 | 1644437 | 28.2 | 1.00 |
| v*1924            | 381225 | 60.3 | 1.00 | 1746729 | 41.7 | 1.00 | 1374232 | 515.7 | 0.99 | 1547351 | 29.5 | 1.00 |
| v*1944            | 5138990 | 373.9 | 1.00 | 3578804 | 221.8 | 1.00 | 3508563 | 104.1 | 0.99 | 4026873 | 6.3 | 0.77 |
| v*1955            | 2753333 | 89.5 | 1.00 | 1393168 | 65.4 | 1.00 | 10396220 | 1074.0 | 1.00 | 1336078 | 399.1 | 1.00 |
| v*1956            | 2840764 | 114.7 | 1.00 | 1449423 | 70.0 | 1.00 | 13419375 | 1437.0 | 0.98 | 1607320 | 70.0 | 1.00 |
| v*1959            | 2420412 | 118.3 | 1.00 | 592281 | 29.9 | 1.00 | 11434248 | 1377.2 | 0.69 | 5428577 | 26.4 | 1.00 |
| unif04-52†        | >10^7   | n/a  | 0.28 | 2465882 | 5.2 | 0.79 | >10^7   | n/a  | 0.29 | 4079329 | 5.2 | 0.82 |
| unif04-62†        | 129042 | 1.2 | 1.00 | 543814 | 0.6 | 1.00 | 3140198 | 3.1 | 0.90 | 442513 | 5.2 | 0.82 |
| unif04-65†        | >10^7   | n/a  | 0.48 | 1110469 | 1.3 | 1.00 | 3800951 | 3.7 | 0.84 | 9300797 | 3.7 | 0.84 |
| unif04-80†        | 5433833 | 4.7 | 0.68 | 2017670 | 2.4 | 0.96 | >10^7   | n/a  | 0.34 | 2105533 | 2.8 | 0.94 |
| unif04-83†        | >10^7   | n/a  | 0.04 | 5206203 | 6.3 | 0.77 | >10^7   | n/a  | 0.24 | 5879298 | 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†        | 1426062 | 1.6 | 0.97 | 538086 | 0.7 | 1.00 | 2634841 | 2.9 | 0.91 | 572947 | 0.8 | 1.00 |
| unif04-99†        | >10^7   | n/a  | 0.32 | 4901045 | 5.0 | 0.87 | >10^7   | n/a  | 0.34 | 3503253 | 5.2 | 0.81 |
| O*1582            | 1503245 | 176.6 | 0.98 | 1125078 | 159.2 | 1.00 | >10^7   | n/a  | 0.34 | 10819125 | 162.6 | 0.99 |
| O*1584            | 4311571 | 51.0 | 1.00 | 3628154 | 50.6 | 1.00 | 5949093 | 5605.5 | 0.69 | 3710475 | 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            | 20410780 | 242.3 | 0.96 | 10724273 | 155.8 | 1.00 | >10^8   | n/a  | 0.38 | 11514246 | 174.3 | 0.99 |
| O*1586            | 1911213 | 222.9 | 0.94 | 1564195 | 225.5 | 0.99 | >10^8   | n/a  | 0.27 | 1453893 | 249.3 | 0.99 |
| O*1587            | 1692114 | 18.8 | 1.00 | 1206202 | 17.9 | 1.00 | 12099223 | 2846.3 | 0.96 | 1426990 | 21.3 | 1.00 |
| O*1588            | 19823423 | 242.1 | 0.97 | 1607353 | 228.3 | 1.00 | >10^8   | n/a  | 0.36 | 1440395 | 227.0 | 0.99 |
| O*1589            | 7727511 | 90.9 | 1.00 | 4813256 | 66.6 | 1.00 | 5738374 | 10081.0 | 0.50 | 5031016 | 75.3 | 1.00 |

We compare the performance of Hybrid with γ=10 (the default value), adaptG^2WSAT_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.raison.anu.edu.au/~anbu/. From these two tables, we summarize the strengths of the performance of Hybrid.

1. Among the 3 algorithms adaptG^2WSAT_P, VW, and R+adaptNovelty+, adaptG^2WSAT_P exhibits the best performance on parity, the industrial instances, and the 2 groups of random instances. Hybrid inherits the strengths of adaptG^2WSAT_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^2WSAT_P on the following 6 groups: blockworld, Beijing, QG, SSS.1.0a, the crafted instances, and the industrial instances. Hybrid outperforms VW on the following 8 groups: blockworld, Beijing, GCP, parity, QG, the industrial instances, and the 2 groups of random instances. Hybrid outperforms R+adaptNovelty+ on the following 7 groups: blockworld, 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^2WSAT_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 $\gamma$, the default value 10. In contrast, adapt$^2$W SAT$_P$, 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 adapt$^2$W SAT$_P$, 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 adapt$^2$W SAT$_P$, 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 adapt$^2$W SAT$_P$, VW, Hybrid ($\gamma=10$) and Hybrid_opposite ($\gamma=10$) on structured and crafted instances. In Hybrid ($\gamma=10$) and Hybrid_opposite ($\gamma=10$), $s=0$.

|       | adapt$^2$W SAT$_P$ | VW | Hybrid ($\gamma=10$) | Hybrid_opposite ($\gamma=10$) |
|-------|-------------------|----|----------------------|-------------------------------|
|       | #flips | time | suc | #flips | time | suc | #flips | time | suc | #flips | time | suc |
| qg1-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg2-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg3-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg4-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg5-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg6-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg7-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg8-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| qg9-9 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |

The chart represents the performance of the algorithms on various instances, with $\gamma=10$.
Table 10. Experimental results for adaptG\(^2\)WSAT\(_P\), 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\(^2\)WSAT\(_P\) | VW | Hybrid (γ=10) | Hybrid\_opposite (γ=10) |
|---------|-------------------------|----|-------------|------------------------|
|         | #flips | time | suc       | #flips | time | suc       | #flips | time | suc       |
| v*1912  | 3419845 | 101.6 | 1.00 | 6152592 | 3037.7 | 0.68 | 3570353 | 95.1 | 1.00 | 1774649 | 2103.1 | 0.77 |
| v*1915  | 11570030 | 372.6 | 1.00 | > 10\(^8\) | n/a | 0.18 | 11994448 | 999.1 | 1.00 | > 10\(^8\) | n/a | 0.14 |
| v*1923  | 1300054 | 31.1 | 1.00 | 1258563 | 428.5 | 0.99 | 1440447 | 28.2 | 1.00 | 8674682 | 367.8 | 1.00 |
| v*1924  | 1746729 | 41.7 | 1.00 | 1374432 | 515.7 | 0.99 | 1537351 | 35.1 | 1.00 | 2814347 | 503.8 | 1.00 |
| v*1944  | 3587804 | 221.8 | 1.00 | 8554145 | 7971.7 | 0.69 | 3508563 | 194.1 | 1.00 | 99909890 | 6607.3 | 0.74 |
| v*1955  | 1393168 | 65.4 | 1.00 | 10396220 | 1074.0 | 1.00 | 1336078 | 59.9 | 1.00 | 10218659 | 6017.3 | 1.00 |
| v*1956  | 1494423 | 70.0 | 1.00 | 14393755 | 1437.0 | 0.98 | 11351370 | 1096.5 | 1.00 | 15883537 | 1903.1 | 0.77 |
| v*1959  | 597281 | 29.9 | 1.00 | 11443442 | 1377.2 | 1.00 | 5428327 | 26.4 | 1.00 | 1135528 | 1101.3 | 1.00 |

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\(^2\)WSAT\(_P\) 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 uneven, and selects a variable to flip according to heuristic adaptG\(^2\)WSAT\(_P\) 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\(^2\)WSAT\(_P\). Hybrid\_random selects a heuristic from heuristic VW and heuristic adaptG\(^2\)WSAT\(_P\) 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\(^2\)WSAT\(_P\) and VW. Specifically, Hybrid\_opposite inherits the poor performance of adaptG\(^2\)WSAT\(_P\) 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^2W SATp, VW, Hybrid (γ=10),and Hybrid_random on the structured and crafted instances. In Hybrid (γ=10) and Hybrid_random, s = 0.

| adaptG^2W SATp | VW | Hybrid (γ=10) | Hybrid_random |
|----------------|----|---------------|---------------|
| #flips | time | suc | #flips | time | suc | #flips | time | suc |
| **Random** | | | | | | | | |
| 304756 | 22.5 | 0.66 | 597473 | 2.5 | 0.99 | 376008 | 1.6 | 1.00 |
| 186839 6.0 | 1.00 | | 184557 | 3.0 | 1.00 | | | |
| **Random** | | | | | | | | |
| 1398701 | 16.1 | 0.92 | 47090 | 2.7 | 1.00 | 89587 | 2.6 | 1.00 |
| 555475 | 3.1 | 1.00 | 54245 | 2.6 | 1.00 | 116633 | 2.8 | 1.00 |
| **Random** | | | | | | | | |
| 408308 | 16.5 | 0.88 | 42881 | 2.5 | 1.00 | 99759 | 2.8 | 1.00 |
| 520705 | 3.6 | 1.00 | 605296 | 16.5 | 0.88 | 42881 | 2.5 | 1.00 |
| **Random** | | | | | | | | |
| 436271 | 3.1 | 1.00 | 35079 | 2.4 | 1.00 | 79277 | 2.1 | 1.00 |
| g250-29 | > 10^7 | n/a | > 10^7 | n/a | 0.18 | 1036322 | 94.0 | 0.88 |
| log^2-cl | 94364 | 0.1 | 1.00 | 190938 | 0.1 | 1.00 | 13543 | 0.1 | 1.00 |

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

| adaptG^2W SATp | VW | Hybrid (γ=10) | Hybrid_random |
|----------------|----|---------------|---------------|
| #flips | time | suc | #flips | time | suc | #flips | time | suc |
| **Random** | | | | | | | | |
| 31500 | 504.5 | 0.58 | 53878 | 258.5 | 0.58 | 35382 | 212.4 | 0.50 |
| 106603 | 207.1 | 0.58 | 106931 | 212.4 | 0.50 | 35382 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 185826 | 278.2 | 0.58 | 25444 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| 1494223 | 70.0 | 1.00 | 25444 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 59928 | 212.4 | 0.50 | 59928 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| 1115038 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 128142 | 212.4 | 0.50 | 128142 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| 592859 | 212.4 | 0.50 | 592859 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 430146 | 212.4 | 0.50 | 430146 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| 113265 | 212.4 | 0.50 | 113265 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 1357310 | 70.0 | 1.00 | 1357310 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
| 257017 | 212.4 | 0.50 | 257017 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 109453 | 212.4 | 0.50 | 109453 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| 1190864 | 70.0 | 1.00 | 1190864 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 1251856 | 428.5 | 0.99 | 1251856 | 428.5 | 0.99 | 106931 | 212.4 | 0.50 |
| large.c | 11520 | 212.4 | 0.50 | 11520 | 212.4 | 0.50 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 2851076 | 278.2 | 0.58 | 2851076 | 278.2 | 0.58 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 1154538 | 70.0 | 1.00 | 1154538 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
| 1154538 | 70.0 | 1.00 | 1154538 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
| **Random** | | | | | | | | |
| 3157783 | 70.0 | 1.00 | 3157783 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
| 1154538 | 70.0 | 1.00 | 1154538 | 70.0 | 1.00 | 106931 | 212.4 | 0.50 |
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 adaptG2WSATp 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 adaptG2WSATp 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. Stützle. 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.
Switching Criterion in Local Search for SAT

[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. Brügge. 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 Satisfiability Problems. In *Proceedings of AAAI-2000* (2000), 310–315.