

A Method to Avoid Duplicative Flipping in Local Search for SAT

Thach-Thao Duong, Duc Nghia Pham, and Abdul Sattar

Queensland Research Laboratory, NICTA
and Institute for Integrated and Intelligent Systems, Griffith University,
QLD, Australia
{thao.duong,duc-nghia.pham,abdul.sattar}@nicta.com.au

Abstract. Stochastic perturbation on variable flipping is the key idea of local search for SAT. Observing that variables are flipped several times in an attempt to escape from a local minimum, this paper presents a duplication learning mechanism in stagnation stages to minimise duplicative variable flipping. The heuristic incorporates the learned knowledge into a variable weighting scheme to effectively prevent the search from selecting duplicative variables. Additionally, probability-based and time window smoothing techniques are adopted to eliminate the effects of redundant information. The integration of the heuristic and gNovelty⁺ was compared with the original solvers and other state-of-the-art local search solvers. The experimental results showed that the new solver outperformed other solvers on the full set of SAT 2011 competition instances and three sets of real-world verification problems.

1 Introduction

Stochastic local search (SLS) for Satisfiability (SAT) problems is currently a well-studied and developed research field. Many practical problems (e.g. hardware verification, planning, circuit design) modeled under SAT format have been efficiently solved by local search. Since the prior work of GSAT algorithm [13], there has been a tremendous research on improving local search for SAT such as AdaptNovelty⁺ [3], G2WSAT [6], VW2 [10], gNovelty⁺ [9], Hybrid [18], TNM [17], Sparrow [1]. While local search is an emerging approach for solving large random instances, it still struggles to solve structured problems because it is mainly based on perturbation and stochastic process of variable flipping [19]. Therefore, minimising duplicative flipping is an important and a challenging issue that need to be addressed.

There are some techniques to improve the performance of local search such as random walk, noise scheme and clause weighting. Random walk technique [8] randomly selects a variable in an unsatisfied clause to escape local minima. This method is significantly developed to the well-known Novelty⁺ and AdaptNovelty⁺ [3] methods, which controlled by a probability noise. The probability noise scheme of Novelty⁺ improves the diversification of the search process by preferring the second best variable within a diversification noise. While

Novelty⁺ retains a constant diversification noise, AdaptNovelty⁺ adjusts the noise automatically during the search. Clause weighting schemes originated from the work of [12] and have been developed substantially through a series of clause weighting local search algorithms (e.g. DLM [19], SAPS [5], PAWS [15]). In recent SAT competitions, gNovelty⁺, TNM and Sparrow2011 were respectively the winners of the Random category in 2007, 2009, and 2011. gNovelty⁺ is a hybrid of a clause weighting scheme from PAWS, Tabu heuristic and AdaptNovelty⁺. TNM is a combination of G2WSAT and two trap escape heuristics switched by a variable weighting scheme. Sparrow2011 is based on gNovelty⁺ framework and utilizes its own probability-based scoring function in the stagnation stages instead of applying AdaptNovelty⁺.

Due to the fact that local search is based on the perturbation of flipping variables to find solutions, flipping the same variables several times in a short period of search steps especially at stagnation stages is a critical drawback of local search. Especially during stagnation stages, some variables are flipped several times in order to escape local minima. This situation is redundant and time-consuming. We call the circumstance that a variable is flipped more than once within a short period at stagnation stages *duplicative flipping*. Currently, there are some methods to reduce duplicative flipping (e.g. Tabu and variable weighting [10]). Tabu search [14] prevents the search from selecting recent variables within a particular tenure. It strictly prohibits duplicative flipping within a given tenure. In addition, variable weighting schemes are used in VW2 [11], Hybrid [18] and TNM [17] for diversification boosting. VW2 employs variable weights to count the number a variable flipped. By preferring the least often selected variables, it aims to minimize duplicative flipping for better diversification in the entire search process. In contemporary local search methods, there is an absence of study on repetitively flipping variables during stagnation stages.

This paper presents a method to minimise duplicative flipping at stagnation stages during the search procedure. The structure of this paper is divided into 4 parts. The introduction provides an overview of contemporary local search methods and their issues. In section two, the approach of duplication avoidance strategy and forgetting techniques are presented. Forgetting techniques (e.g. probability-based smoothing and time window release) are applied to enhance the performance of the heuristic. Afterwards, gNovelty⁺DL, the integration of the heuristic and the gNovelty⁺ framework, is clearly explained. Section three reports experiments on some structure test suite and SAT 2011 competition dataset. Finally, section four sums up our conclusions and future work.

2 Duplication Learning Strategy

2.1 Motivation

Avoidance of duplicative flipping is one important issue in local search algorithms for SAT. To investigate duplicative flipping on structured and random instances, we conducted an experiment to compute the duplication ratio in stagnation paths. We define a *stagnation path* as a sequence of consecutively flipped

variables back-tracking from a local minimum, and *duplicative variables* are variables that appear more than once within a stagnation path. The *duplication ratio* is the fraction of the number of duplicative variables over the number of variables in stagnation paths. In order to limit collective data for study, a parameter tenure k specifies the length of a stagnation path.

Table 1. Average of duplication ratios on verification benchmarks and SAT 2011 Competition dataset. Stagnation paths have the same length of $k=20$.

Instances	vw2	gNovelty ⁺	Sparrow2011
cbmc.test (39)	7%	38%	27%
swv.test (75)	4%	18%	16%
sss-sat-1.0 (100)	11%	50%	74%
Application (149)	2%	14%	4%
Crafted (150)	24%	30%	22%
Random Large (200)	3%	38%	6%
Random Medium (400)	20%	68%	29%

Table 1 shows the average of duplication ratios of VW2, gNovelty⁺ and Sparrow2011. Experiments were conducted on the verification benchmarks : cbmc, swv, sss-sat-1.0 and the SAT 2011 competition dataset. The experiments on verification benchmarks were conducted 50 times for each instance. SAT 2011 competition dataset experiments were run 10 times for each instance. As seen in Table 1, VW2 had the least duplication ratio compared to gNovelty⁺ and Sparrow2011. This was because VW2 employed variable weighting schemes to avoid duplicative flipping. However, variable weights accumulate information of flipping frequencies during the entire search process rather than specifically during stagnation stages.

According to Table 1, the duplication ratios were high during stagnation stages. This phenomenon increases the duration of the search process by selecting the same variables several times. Additionally, duplicative flipping probably leads to previous local minima or previously visited areas. We hypothesize that variables occurring repeatedly in stagnation paths are sensitive and critical. These variables should not be selected as they are likely to cause stagnation and duplicative flipping in the future. In contemporary literature, there are a few methods that have exploited the duplication information at visited stagnation areas when selecting variables for the next steps. Although Tabu search and variable weighting are methods to minimise duplicative flipping, they do not have a learning mechanism to learn from duplicative variables leading to local minima. In our opinion, there remains a need for an efficient method for learning about duplication in stagnation stages and actively preventing the search from selecting these variables. From that point of view, in the following section we propose a heuristic that combines the advantages of the variable weighting and the duplication learning heuristic.

2.2 Duplication Avoidance Heuristics

As described in the previous section, variables that appear in the bounded stagnation paths are considered important to the learning mechanism. The proposed duplication learning (DL) aims to assist local search algorithms in intelligently avoiding variables that are highly potential to cause duplication in the future by exploiting duplication information at previous stagnations. We associate a duplication weight to each variable and these weights are initially set to 0. At each local minimum, duplication weights of duplicative variables in the stagnation paths are increased by one. Hence, a high value of duplication weight indicates potential likelihood of causing duplicative flipping in trap areas if the variable is chosen to be flipped. In other words, variables whose duplication weight are high have more conflicts with the current solution assignment. For that reason, variables having low duplication weights are preferred to be selected in the next steps. The use of duplication weights and tenure k are inspired by the VW2 algorithm [10] and Tabu search [14]. Compared to VW2, however the weight updating procedure is applied at the stagnation stages and the heuristic does not completely forbid variables in the tenure like Tabu heuristic.

Algorithm 1 describes the proposed duplication learning heuristic in combination with two forgetting techniques. The history of flipped variables during search progress are stored in stack H . Lines 3 - 5 of Algorithm 1 describe the way the stagnation paths are popped from the stack of flipped variables H . At a stagnation stage, variables duplicately flipped in the stagnation path have their weights increased by one (lines 7 - 9). In order to investigate the effectiveness of the learning mechanism, the two forgetting techniques (e.g. the time window release and probability-based smoothing techniques) are applied in duplication prevention heuristic (lines 11 - 19 Alg. 1). These two techniques are explained clearly in the following sections. The parameters time window queue W and its size T are used for time window release strategy; and the duplication smooth probability dp is used for probability-based smoothing mechanism.

2.3 Techniques to Forget Learned Information

As the search visits a large number of local minima, duplication weights may not correctly reflect the duplicative scenarios that are relevant to the current location. It is necessary to apply a mechanism to reduce over-learned duplication weights. For that reason, we integrate a probability-based smoothing technique and a time window mechanism to the heuristic.

Probability-Based Smoothing. The probability-based smoothing mechanism is a simple smoothing technique controlled by a probability parameter dp . Within the smoothing probability dp , duplication weights are reduced by one (Alg. 1 lines 11 - 16). The aim of the probability-based smoothing technique is to reduce over-valued duplication weights which are mostly irrelevant information for selecting variables.

Algorithm 1: Duplication Learning Strategy $DL(k, T, dp, H, W)$

Input : tenure k , flipped variable history stack H , time window queue W , time window size T , duplication smooth probability dp

```

1  stagnation_path =  $\emptyset$  ;
2  duplication_list =  $\emptyset$  ;
3  for  $i \leftarrow 1$  to  $T$  do
4     $var \leftarrow pop\_stack(H)$  ;
5    stagnation_path  $\leftarrow var$  ;
6  end
7  for all  $var$  appears more than one time in stagnation_path do
8    duplication_weight[ $var$ ] ++ ;
9    duplication_list  $\leftarrow var$  ;
10 end
11 if  $T > 0$  then
12   // Using time window release
13   if  $Size(W) == T$  then
14     further_duplication_list =  $pop\_queue(W)$ ;
15     for all  $var$  in further_duplication_list do duplication_weight[ $var$ ] -- ;
16     remove further_duplication_list from  $W$ ;
17    $W \leftarrow push\_queue(W, duplication\_list)$ ;
18 if  $dp > 0$  then
19   // Probability-based Smoothing
20   if within probability of  $dp$  then
21     for all  $var$  if duplication_weight[ $var$ ]  $> 1$  do duplication_weight[ $var$ ] -- ;
22   end
23 end

```

Time Window Release. The time window release technique (lines 17 - 19 Alg. 1) resembles a window that moves along-side the search in progress. The aim of this technique is to restrict effects of the duplication weights into limited recent stagnation times. To elaborate, the learned information within the window area is accumulated in duplication weights while the information outside the window is discarded. We define a *duplication list* as a list of duplicative variables occurred in a stagnation path. The time window queue W is an ordered queue of T most recent duplication lists. The size of W are bounded by parameter T . Once a stagnation path is released from the window view, its learned experience has elapsed. In other words, the duplication weights of variables in the corresponding duplication list are decreased by one. The further duplication list is removed from the window and a newly encountered duplication list is pushed into the queue.

2.4 gNovelty⁺DL: Integration of Duplication Learning into gNovelty⁺

This section describes the integration of duplication learning heuristic into the gNovelty⁺ algorithm. gNovelty⁺ uses a clause weighting scheme while duplication learning employs a variable weighting mechanism. Therefore, gNovelty⁺DL, an integration of duplication learning into gNovelty⁺, is a combination of clause weighting and variable weighting focused on duplication avoidance. Algorithm 17 presents the pseudo-code of gNovelty⁺DL.

As described in [9], gNovelty⁺ has two parameters: a random walk probability wp and a smooth probability sp . The DL strategy requires three more parameters which are the tenure k , the time window size T and the duplication smoothing probability dp . The parameter dp is used for the probability-based smoothing strategy described in section 2.3. In the initialization phase, duplication weights of variables are set to zero. The variable history H and the time window W are initialized empty. Because gNovelty⁺ is a clause weighting local search in which the score function is computed based on clause weights, local minima are considered at the steps when there is no promising variable¹. In this circumstance, the DL strategy is applied (line 11). The criteria of selecting variables is slightly different to gNovelty⁺. Instead of breaking tie the least recent flipped variable, the heuristic breaks tie by the lowest duplication weight. In lines 8 and 12, the algorithm selects the most promising variable in terms of score value, and then breaks tie by lowest duplication weights. Finally, the selected variable is pushed into the history stack H .

Algorithm 2: gNovelty⁺DL(k, T, dp, sp)

Input : A formula Θ , random walk probability $wp = 0.01$, smooth probability sp , tenure k , time window size T , duplication smoothing probability dp
Output: Solution α (if found) or TIMEOUT

```

1 randomly generate a candidate solution  $\alpha$ ;
2  $W = \emptyset$ ;  $H = \emptyset$ ;
3 initialize duplication_weights to 0;
4 while not time out do
5   if  $\alpha$  satisfies the formula  $\Theta$  then return  $\alpha$  ;
6   if within the probability  $wp$  then Random walk;
7   else if there are promising variables then
8     | select the most promising variable, breaking tie by lowest duplication_weight;
9   else
10    | update (and smooth in probability  $sp$ ) clause weights;
11    | // Stagnation happens: perform duplication learning
12    | DL( $k, T, dp, H, W$ );
13    | AdaptNovelty+: (breaks tie by lowest duplication_weight);
14  end
15  flip variable  $var$ ;
16   $H \leftarrow push\_stack(H, var)$ ;
17 end
18 return TIMEOUT;

```

3 Experiments

In these experiments, gNovelty⁺DL was dispersed into 4 variants differentiated by the combination use of the time window release and probability-based smoothing. The list of gNovelty⁺ variants is shown in Table 2. The intention of these variants was to examine the feasibility of DL heuristic as well as effects of forgetting techniques. By adding both time window release and probability-based smoothing in gNovelty⁺Rws, we aim to investigate the capabilities of the two forgetting techniques. In the preliminary experiment, the performance of

¹ A promising variable is one that improves the objective score [9].

gNovelty⁺Rp is examined against some common SLS solvers on ternary chains to test the diversification capability. Further experiments were conducted on structured verification benchmarks and the full SAT 2011 competition dataset and compared gNovelty⁺DL with gNovelty⁺ (uses clause weighting), VW2 (uses variable weighting), and Sparrow2011 (the winner of the latest SAT competition). Table 5 presents the parameter settings for gNovelty⁺DL variants. These parameter configurations were optimized by ParamILS [4], a local search optimization tool for parameterized algorithms. The experiments were conducted on the Griffith University’s Gowonda HPC Cluster Intel(R) Xeon(R) CPU X5650 2.67GHz.

Table 2. Variants of gNovelty⁺DL heuristic

Time window release	Probability-based smoothing	
	No	Yes
No	Rp	Rps
Yes	Rw	Rws

3.1 Preliminary Experiments on Ternary Chains

Ternary chain is an artificial problem that deliberately simulates chains of variable dependency in structure problems. It is employed to study the diversification capability of local search algorithms in [10], due to the fact that it has only one solution that assigns all variables to true. Therefore, flipping any variable from true to false is likely to move away from the solution. Given these characteristics, local search algorithms will easily become trapped. The following is the formulation of a ternary chain with N variables.

$$(x_1) (x_2) (x_1 \wedge x_2 \rightarrow x_3) \dots (x_{N-2} \wedge x_{N-1} \rightarrow x_N)$$

Fig 1 depicts a comparison between gNovelty⁺DL with some popular solvers (e.g. rots - Tabu search [14], adaptG2WSAT [7], PAWS [15], VW2 [11], TNM [17], gNovelty⁺ [9], Sparrow2011 [1]). It was apparent that VW2, gNovelty⁺ and gNovelty⁺Rp performed strikingly better than other solvers. gNovelty⁺ and VW2 could solve ternary chains successfully due to variable weighting and clause weighting respectively. As a result, gNovelty⁺DL, which took advantage of both gNovelty⁺ and VW2, was an improvement over these two algorithms. This indicated that gNovelty⁺Rp had better diversification than the other solvers in this experiments.

3.2 Experiments on Structural Benchmarks

The real-world problem sets we selected for this experiment are cbmc, swv and sss-sat-1.0. The first two instance sets were software verification problems: (i) 39 cbmc instances generated by a bounded model checking tool and (ii) 75 swv

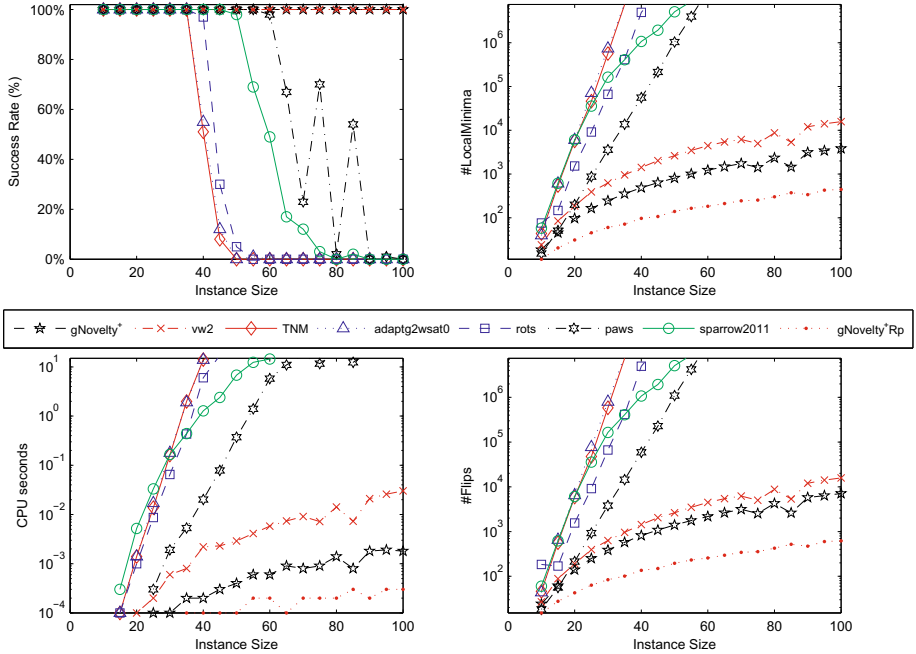


Fig. 1. Experiments on Ternary Chains: size ranging from [10,100]. Based on 100 runs where each run had a 20-second timeout. Methods are compared in terms of success rate, CPU time, the number of local minima and the number of flips. gNovelty+Rp used tenure $k = 20$.

instances generated by the CALYSTO checker ². The third one was Velev’s sss-sat-1.0 containing 100 instances of encoding verification of super-scalar micro-processors ³. Even though these instance sets can be easily solved by PICOSAT [2] (a systematic search algorithm), they are currently a remarkable challenges for SLS solvers. Table 3 shows the evaluation of gNovelty+DL variants with gNovelty+, VW2 and Sparrow2011 in terms of success rates, average of median CPU times and number of duplication flipping in thousands of stagnation paths.

Generally speaking, all gNovelty+DL variants performed consistently better than other solvers across the three instance sets. Especially on cbmc and sss-sat-1.0 dataset, the success rate of gNovelty+DL was raised to 100%, which was a significant improvement over the other algorithms. Furthermore, the success rate of swv instances was doubled to 48% compared with other solvers. It was discovered that 50% of swv instances have not been solved consistently by SLS-based solvers [16]. Moreover, all gNovelty+DL variants significantly decreased

² The test instances of cbmc and swv are available at <http://people.cs.ubc.ca/davet/papers/sat10-dave-instances.zip>

³ Available at http://www.miroslav-velev.com/sat_benchmarks.html

Table 3. Experiments on cbmc, swv, sss-sat-1.0: based on 50 runs with a timeout of 600 seconds per run. Each cell presents the success rate, the average of median CPU times in seconds and the average of duplication times in thousands.

Instances	VW2	Sparrow2011	gNovelty ⁺	gNovelty ⁺ DL			
				Rp	Rps	Rw	Rws
cbmc (39)	31%	51%	85%	100%	100%	100%	100%
	439.128	384.359	247.997	1.659	2.343	1.452	1.774
	6, 845	231	3, 886	74	69	40	72
swv (75)	23%	21%	25%	48%	47%	48%	44%
	466.002	486.949	459.097	335.343	348.875	327.477	350.927
	280	59	1, 995	6, 036	232	1, 023	2, 903
sss-sat-1.0 (100)	24%	7%	50%	100%	100%	100%	100%
	481.357	572.993	348.512	2.409	2.729	2.241	4.376
	6, 700	1, 065	2, 177	293	139	212	522

duplication flipping times in stagnation paths compared with other algorithms. However, there is no clear difference in terms of success rate and CPU times among gNovelty⁺DL variants.

3.3 Experiments on the SAT 2011 Competition Dataset

The experiment on the SAT 2011 competition dataset was a comparison between gNovelty⁺DL, Sparrow2011, gNovelty⁺ and VW2. Figure 2 plots the performance of the best gNovelty⁺DL variant with Sparrow2011 in 3 categories : Application & Crafted (the structure instances), Random (problems created by randomization), and All (the full SAT 2011 competition dataset). As illustrated in Figure 2, gNovelty⁺ did not perform as well as Sparrow2011 on the Random category, but it was slightly better than Sparrow2011 in the Application & Crafted category. With the boost from the DL heuristic, gNovelty⁺DL improved the performance of gNovelty⁺ on both the Application & Crafted and Random categories. It is evident that the duplication learning heuristic assisted the gNovelty⁺ in overcoming the limitations on both structure instances and random instances.

Table 4 shows the performance of the algorithms in terms of the success rates and the averages of the minimum, median, maximum CPU times. The data were split into 4 groups ⁴: (i) Application: 8 instances; (ii) Crafted: 82 instances; (iii) Random Large: 64 instances (iv) Random Medium: 201 instances. For all categories, most gNovelty⁺DL variants performed better than other solvers. There were exceptions in the case of Random Large, e.g. gNovelty⁺Rp (9%) and gNovelty⁺Rps (17%) had worse results than Sparrow2011 (20%).

⁴ Because there were many instances cannot be solved at any run, we decided to report instances solved in least one run by any solver. Therefore, the number of instances in Table 4 does not reflect the real number of instances in the original SAT 2011 competition data set.

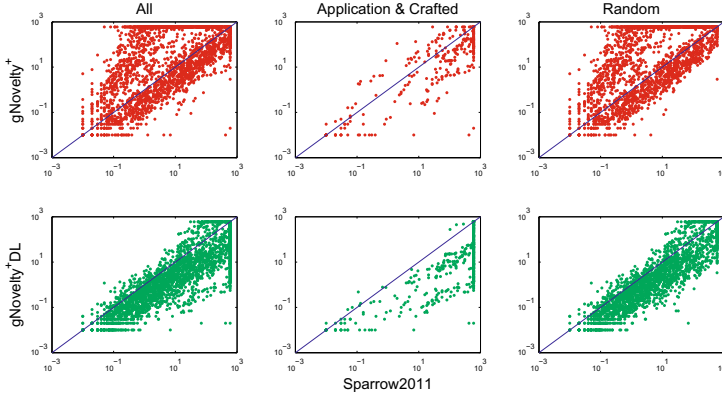


Fig. 2. Experiments on SAT 2011 Competition Benchmark: a comparison of CPU run times between $gNovelty^+DL$ and Sparrow2011

Table 4. Experiments on SAT 2011 Competition dataset: based on 10 runs and a timeout of 600 seconds per run. The reported rows are success rates, the averages of the minimum, median, and maximum CPU times (in seconds).

Instances	VW2	Sparrow2011	$gNovelty^+$	$gNovelty^+DL$			
				Rp	Rps	Rw	Rws
Application	12%	25%	12%	62%	38%	38%	62%
(8)	526.239	461.824	464.149	131.018	130.770	55.882	183.981
	526.838	515.799	527.431	298.067	397.774	389.998	333.106
	527.603	543.695	527.503	426.520	463.189	483.426	456.875
Crafted	37%	56%	62%	87%	82%	87%	82%
(82)	373.386	237.720	202.872	71.112	53.848	62.977	82.665
	390.427	285.159	251.332	111.388	128.326	110.367	129.848
	402.716	338.434	285.784	157.880	170.066	146.968	161.738
Random	0%	20%	0%	9%	27%	31%	17%
Large (64)	600.007	437.202	576.600	426.179	384.403	334.614	417.429
	600.010	509.487	600.000	566.617	525.684	509.716	550.516
	600.010	555.076	600.000	594.413	591.152	579.286	589.425
Random	19%	100%	75%	96%	99%	100%	99%
Medium (201)	428.007	5.049	130.371	4.017	4.655	2.169	2.427
	499.761	40.671	184.698	38.837	21.945	13.392	21.432
	527.371	99.540	241.121	102.376	55.434	48.654	68.034

Comparing amongst $gNovelty^+DL$ variants, $gNovelty^+Rp$ was the best performer in the Application & Crafted category while $gNovelty^+Rw$ performed best in Random category.

In Random categories (both medium and large), applying at least one forgetting technique improved the performance. For example, in terms of success rate, $gNovelty^+Rws$ performed better than $gNovelty^+Rp$ in Random Medium categories. However, utilizing both probability-based smoothing and time window

Table 5. gNovelty⁺DL parameters settings

Dataset	Rp		Rw			Rps			Rws			
	<i>k</i>	<i>sp</i>	<i>k</i>	<i>T</i>	<i>sp</i>	<i>k</i>	<i>dp</i>	<i>sp</i>	<i>k</i>	<i>T</i>	<i>dp</i>	<i>sp</i>
cbmc	30	0.4	15	100	0.4	30	0.25	0.4	20	300	0.05	0.4
swv	30	0	15	250	0.05	20	0.2	0.2	30	250	0.4	0.05
sss-sat-1.0	10	0.05	20	200	0.05	10	0.1	0.05	25	200	0.05	0.05
SAT 2011	30	0	25	250	0	10	0.25	0	15	200	0.25	0

release degraded the performance (e.g gNovelty⁺Rws did not perform as good as gNovelty⁺Rw and gNovelty⁺Rs). On large random and medium random, using probability-based smoothing alone improved the results (e.g in Random Large category gNovelty⁺Rps boosted gNovelty⁺Rp from 9% to 26%). However, probability-based smoothing combined with time window decreased the results of time window (for instance in Random Large category, gNovelty⁺Rws degraded the performance of gNovelty⁺Rw from 31% to 17%).

In contrast with Random categories, applying only one forgetting technique did not improve performance in the Application category. For example, the success rates of gNovelty⁺ Rps and gNovelty⁺ Rw were just 38% compared with 62% for gNovelty⁺Rp. In the Crafted category, the results of gNovelty⁺DL variants were not significantly different and they all gained better performance than the other algorithms.

Overall, for structured instances, using forgetting techniques is not recommended because it reduces learned information. In contrast, applying time window release (e.g gNovelty⁺Rw) performed best for random instances. This is due to random instances being more loosely constrained than structure instances. Hence, it is necessary to retain learned information over a limited period of stagnation times for random instances.

4 Conclusion and Future Work

In summary, the duplication learning was feasible and robust for the considered benchmarks when applied to the gNovelty⁺ framework. gNovelty⁺DL performed well on both structured and random instances compared with contemporary SLS solvers. In addition, it was proved to reduce duplication and help search converged quickly to solutions. It can be concluded that duplication learning is a reasonably alternative guidance for local search for the purpose of duplication avoidance.

In addition, the effect of forgetting techniques were examined. Generally, forgetting techniques should not be applied to structured instances. In contrast, to solve random instances efficiently, it is better to adopt forgetting techniques to reduce over-learned stagnation information. To further clarify, gNovelty⁺Rp was a good choice for structured benchmarks while the gNovelty⁺Rw was a good choice for random instances. However, applying both probability-based smoothing and a time window mechanism degraded the performance. Evidently, forgetting techniques are efficient on loosely constrained problems.

According to these results, a further investigation into duplication avoidance using learning mechanisms is worthwhile. Indeed, it is necessary to develop a

more intelligent learning strategy and parameter adaptation mechanism. Moreover, in order to examine the generalization of the heuristic, its integration into different SLS solvers should be carefully studied.

References

1. Balint, A., Fröhlich, A.: Improving Stochastic Local Search for SAT with a New Probability Distribution. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 10–15. Springer, Heidelberg (2010)
2. Biere, A.: Picosat essentials. JSAT 4(2-4), 75–97 (2008)
3. Hoos, H.H.: An adaptive noise mechanism for WalkSAT. In: Proceedings of AAAI 2002, pp. 635–660 (2002)
4. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: Paramils: An automatic algorithm configuration framework. J. Artif. Intell. Res. (JAIR) 36, 267–306 (2009)
5. Hutter, F., Tompkins, D.A.D., H. Hoos, H.: Scaling and Probabilistic Smoothing: Efficient Dynamic Local Search for SAT. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 233–248. Springer, Heidelberg (2002)
6. Li, C.-M., Huang, W.Q.: Diversification and Determinism in Local Search for Satisfiability. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 158–172. Springer, Heidelberg (2005)
7. Li, C.-M., Wei, W., Zhang, H.: Combining Adaptive Noise and Look-Ahead in Local Search for SAT. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 121–133. Springer, Heidelberg (2007)
8. McAllester, D.A., Selman, B., Kautz, H.A.: Evidence for invariants in local search. In: AAAI/IAAI, pp. 321–326 (1997)
9. Pham, D.N., Thornton, J., Grettton, C., Sattar, A.: Combining adaptive and dynamic local search for satisfiability. JSAT 4(2-4), 149–172 (2008)
10. Prestwich, S.: SAT problems with chains of dependent variables. In: Discrete Applied Mathematics, vol. 3037, pp. 1–22 (2002)
11. Prestwich, S.D.: Random Walk with Continuously Smoothed Variable Weights. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 203–215. Springer, Heidelberg (2005)
12. Selman, B., Kautz, H.A.: Domain-independent extensions to gsat: Solving large structured satisfiability problems. In: IJCAI, pp. 290–295 (1993)
13. Selman, B., Levesque, H.J., Mitchell, D.G.: A new method for solving hard satisfiability problems. In: AAAI, pp. 440–446 (1992)
14. Taillard, É.D.: Robust taboo search for the quadratic assignment problem. Parallel Computing 17(4-5), 443–455 (1991)
15. Thornton, J.R., Pham, D.N., Bain, S., Ferreira Jr., V.: Additive versus multiplicative clause weighting for SAT. In: Proceedings of AAAI 2004, pp. 191–196 (2004)
16. Tompkins, D.A.D., Balint, A., Hoos, H.H.: Captain Jack: New Variable Selection Heuristics in Local Search for SAT. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 302–316. Springer, Heidelberg (2011)
17. Wei, W., Li, C.M.: Switching between two adaptive noise mechanisms in localsearch. In: Booklet of the 2009 SAT Competition (2009)
18. Wei, W., Li, C.-M., Zhang, H.: Switching among Non-Weighting, Clause Weighting, and Variable Weighting in Local Search for SAT. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 313–326. Springer, Heidelberg (2008)
19. Wu, Z., Wah, B.W.: Trap escaping strategies in discrete lagrangian methods for solving hard satisfiability and maximum satisfiability problems. In: AAAI/IAAI, pp. 673–678 (1999)