Diversify intensification phases in local search for SAT with a new probability distribution

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

Queensland Research Laboratory, NICTA and
Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia
{t.duong,d.pham,a.sattar}@griffith.edu.au

Abstract. A key challenge in developing efficient local search solvers is to intelligently balance diversification and intensification. This study proposes a heuristic that integrates a new dynamic scoring function and two different diversification criteria: variable weights and stagnation weights. Our new dynamic scoring function is formulated to enhance the diversification capability in intensification phases using a user-defined diversification parameter. The formulation of the new scoring function is based on a probability distribution to adjust the selecting priorities of the selection between greediness on scores and diversification on variable properties. The probability distribution of variables on greediness is constructed to guarantee the synchronization between the probability distribution functions and score values. Additionally, the new dynamic scoring function is integrated with the two diversification criteria. The experiments show that the new heuristic is efficient on verification benchmark, crafted and random instances.

1 Introduction

Stochastic Local Search (SLS) is a competitive and an efficient approach to find the optimal solution or the approximately optimal solution for very large and complex combinatorial problems. Some examples of practical combinatorial problem instances that have been solved efficiently by SLS under the Satisfiability (SAT) framework are hardware verification and planning. Despite this significant progress, SLS solvers still have limitations compared with systematic solvers in practical and structured SAT problems as evident through the series of SAT competitions. Because structured and practical SAT problems have tighter constraints than randomized SAT problems, SLS algorithms are easily trapped in local minima and have difficulty to escape from stagnation. This problem does not exist in systematic search algorithms because of the nature of complete searching strategies.

Since the introduction of the GSAT algorithm [15], there have been huge improvements in developing efficient SLS algorithms for SAT. These improvements need to properly regulate diversification and intensification in local search. There are some common techniques to boost diversification such as random walk [10] and Novelty$^+$ [8]. In addition, some diversification boosting methods includes variable weighting [13] or clause weighting [16, 12, 6]. As reported in [14], the clause weighting scheme is able to escape the local minima by focusing on satisfying long-time unsatisfied clauses. The Hybrid [18] and TNM [17] algorithms exploit variable weight distribution to regulate the two noise heuristics but do not directly exploit variable weights to select variables. Recently, stagnation weights were introduced as a new diversification criterion to avoid local minima in gNovelty$^+$PCL [11]. Stagnation weights can be considered as an extension to variable weights because they record frequencies of variables involved in stagnation paths [5]. Sparrow2011 [1], the winner of the SAT competition 2011, is based on gNovelty$^+$ [12] framework but instead of using the Novelty$^+$ strategy at stagnation phases, it employs its own dynamic scoring function to escape from local minima. Recently, CCASat [3], the winner of SAT Challenge 2012, heuristically switches between two greedy modes and one diversification mode, and uses configuration checking to prevent the blind unreasonable greedy search.

In terms of intensification enhancement, the majority of local search solvers greedily explore the search space. More specifically, a local search will choose the most decreasing variable (i.e. a variable that leads to the most decrease in the number of unsatisfied clauses if being flipped). If there is more than one best decreasing variable, the algorithm prefers the variable with better diversification (i.e the least recently flipped or the lowest variable weight). The diversification mode is invoked when the search cannot greedily explore the search space. During the intensification mode, the scoring function (i.e. objective function) is very important. The drawback of most scoring function and variable selection methods is a lack of compromise between scores and tiebreak criteria. Mostly the tiebreak criteria are variable properties such as variable ages in most SLS solvers and variable weights in VW2 or stagnation weights in gNovelty$^+$PCL. The tiebreak criteria are considered as diversification boosting properties. Despite the fact that the current gradient-based scoring function (i.e. score in G2WSAT) in greedy phases works efficiently with current SLS solvers, a more advanced scoring function is needed to balance the score and the diversification tiebreaks. It motivates us to develop a single scoring function that combines greedy scores and diversification criteria.

In this work, we present a new SLS solver which uses an integration of a new probability-based dynamic scoring function as the objective function and two diversification criteria. The proposed dynamic function is controlled by a diversification noise α and is designed as a combination of clause-weighting score function and diversification criteria. The remainder of this paper is structured as follows. Section 2 summaries the background of SLS in developing objective functions. The motivation and construction of the probability-based dynamic formula are presented in section 3. Section 4 describes our algorithm, named PCF. The experiments on verification, crafted and random instances of SAT competitions are reported in section 5. This section also discusses the coverage of optimal diversification parameters. Section 6 concludes the paper and outlines the future work.

2 Preliminaries

Most modern SLS solvers operate in two modes : greedy (or intensification) mode and random (or diversification) mode. Starting from a randomized candidate solution, the solver computes the objective function for each variable. The function reflects the improvements in regards to the decrease in the number of the number of unsatisfied clauses when a variable is flipped. The most basic objective function (or scoring function) is computed as the decrease in the number of unsatisfied clauses. While this traditional score was used by TNM, sattime2011, the most up-to-date dynamic scoring function is the additive clause-weighting score used by many SLS solvers (e.g. gNovelty$^+$, Sparrow, EagleUP, CCASat).

In the greedy mode, if there exist variables with positive scores, the solver will select the variable with the highest score, breaking ties by the least recently flipped variables. Otherwise, the local search resides in a local minimum in which there is no possible greedy move. In such cases, it selects variables according to the random mode. There are numerous heuristics for the variable selection at the random mode. Most of them randomly pick an unsatisfied clause and select variables within that clause. Random walk is the simplest way of selecting variables randomly from the selected clauses. Novelty$^+$ is the most common and efficient scheme integrated in adaptG2WSAT, gNovelty$^+$.

2.1 Basic scoring function

The most basic scoring function for SAT is proposed by GSAT. It defines the number of unsatisfied clauses. Afterwards, the score is computed in an alternative objective function of the decrease in the number of unsatisfied clauses in G2WSAT. Eq. 1 expresses the score computed in G2WSAT.

$$score(v) = \sum\nolimits_{c} (Cls'(c, \alpha, v) - Cls(c, \alpha)) \tag{1}$$

where $score(v)$ is the decrease in the number of unsatisfied clauses if variable v is flipped. $Cls(c, \alpha)$ is the value of clause c under the candidate solution assignment α. If clause c is satisfied, $Cls(c, \alpha) = 1$, else $Cls(c, \alpha) = 0$. Given an circumstance of assignment α, $Cls'(c, \alpha, v)$ is the value of clause c after variable v is flipped.

2.2 Dynamic scoring function

To prevent the search from getting trapped in local minima, other dynamic penalties of clauses are integrated into the objective function of the search. The SLS using dynamic scoring functions is named the dynamic local search. This method is based on modifying the scoring function at each search step to reevaluate the objective function in conjunction with changed circumstances. The purpose of the dynamic scoring function is to adjust the circumstance of local minima from the static method of computing an objective function. Because the dynamic scoring function can adjust the objective function, it assists the local search to dynamically avoid failing into previous stagnation.

Clause-weighting The state-of-the-art dynamic scoring function is based on the clause weighting scheme. This scheme typically associates weights with clauses. At each step, clause weights are adjusted according to truth value of corresponding clauses. Then instead of minimising the number of false clauses, the algorithms minimise the sum of clause weights. Eq. 2 expresses the clause-weighting score.

$$score_w(v) = \sum\nolimits_c Wgh(c) \times (Cls'(c, \alpha, v) - Cls(c, \alpha)) \tag{2}$$

where $Wgh(c)$ is the clause weight of clause c.

VW2 Another way of computing dynamic scoring function based on variable weights is firstly proposed in VW1 and VW2 [13]. The dynamic scoring function of VW2 uses variable weights as diversification properties involved in the score. The scoring function of VW2 is computed as follows:

$$score_{VW2}(v) = \Big(\sum\nolimits_c break(c, \alpha, v)\Big) + \Big(b \times (vw(v) - \overline{vw})\Big) \tag{3}$$

where $break(c, \alpha, v)$ is set to one if clause c becomes unsatisfied when variable v is flipped in the candidate solution α; otherwise its value is set to zero. $vw(v)$ is the weight of variable v. $\overline{vw}$ denotes the average of variable weights across all variables. b is a pre-defined parameter. The update and continuous smoothing procedure on variable weights vw is described in [13].

Sparrow introduced a dynamic scoring function to overcome local minima. Its scoring function is modeled under a probability distribution and computed based on $score_w$ and variable ages. The variable selection is still based on the $score_w$ in the greedy mode. However, this dynamic scoring function is employed at stagnation phases only. The solver selects randomly one of the yet unsatisfied clauses at random. The selected clause is notated as $u_i = (x_{i_1} \vee .. \vee x_{i_k})$. The probability distribution to select variables in clause u_i is computed as the Eq. 4 .

$$p(x_{i_j}) = \frac{p_s(x_{i_j}) \times p_a(x_{i_j})}{\sum_{l=1}^{k} p_s(x_{i_l}) \times p_a(x_{i_l})} \tag{4}$$

with $p_s(x_{i_j}) = a_1^{score_w(x_{i_l})}$, and $p_a(x_{i_j}) = (\frac{age(x_{i_l})}{a_3})^{a_2}$

The constant a_1, a_2, a_3 are experimentally determined and reported in [1].

3 Probability-based dynamic scoring function

3.1 Motivation

As mentioned in the previous sessions, few solvers have addressed the issue of combining greediness and diversification criteria into a single function. One drawback of variable selections of most SLS solvers is the fact that algorithms greedily select the most promising variable in terms of scores as the first priority. In case two variables have the same score, the algorithms will consider about tiebreak criteria (e.g. variable age). The scores and variable ages are considered separately in variable selection. Moreover, scores have greater priority than variable ages.

In order to construct a single scoring function as a trade-off between scores and tiebreaks, we decided to use probability knowledge to formulate a new dynamic scoring function. The idea of using a probability-based dynamic scoring function was firstly introduced in Sparrow [1]. Sparrow2011 and EagleUP [7] won the first and third places respectively in the SAT 2011 competition in Random track. These two solvers are efficient on random instances, which was attributed to their probability-based scoring function. However, their probability-based scoring function is restricted to stagnation phases whereas the conventional clause-weighting score is still used during intensification phases. We decided to approach the problem differently from the Sparrow formula by using additive formulation opposed to multiplicative formulation (Eq. 4). One reason for creating additive formulation is to modify the function gradually instead of adjusting rapidly as the multiplicative formulation. Additionally, we preferred to employ fewer parameters for users to regulate the scoring function more easily.

3.2 Defining a probability distribution

The proposed dynamic scoring function is formulated as Eq. 5. The scoring function is a summary of greediness and diversification probability distribution. The probability of diversification contribution is regulated by a user-defined parameter α. Thus, greediness probability contribution is regulated by $(1-\alpha)$.

$$P(v_i) = (1-\alpha) \times P_g(v_i) + \alpha \times P_d(v_i) \tag{5}$$

where v_i is the i-th variable and $P(v_i)$ is the probability of selecting variable v_i. The higher the value of $P(v_i)$, the more likelihood v_i is selected. $P_g(v_i)$ is the probability of greediness and $P_d(v_i)$ is the probability of diversification for variable v_i. In accordance to the fact that $P(v_i)$, $P_g(v_i)$, $P_d(v_i)$ are probability distribution functions (pdf), their values are scaled in the range of [0,1].

Probability Distribution on Greediness In this work, we chose $score_w$ in Eq. 2 to compute $P_g(v_i)$. $score_w(v_i)$ is firstly scaled into the range [0,max_{score_w} - min_{score_w}] to satisfy the condition that the probability distribution function is non-negative. The adjusted scoring function for v_i is calculated by Eq. 6.

$$score'_w(v_i) = score_w(v_i) - min_{score_w} \tag{6}$$

where min_{score_w} and max_{score_w} is the minimum and maximum $score_w$ across all variables. Afterwards, the scoring function is normalized to satisfy the condition of a probability distribution function (i.e. probability distribution functions are in the range [0,1] and summarized to one) as follows:

$$P_g(v_i) = \mathcal{N}(score'_w(v_i)) = \frac{score'_w(v_i)}{\sum_j score'_w(v_j)} \tag{7}$$

Probability Distribution on Diversification In this work, we chose two diversification properties separately to compute the probability distribution of diversification $P_d(v_i)$ in order to investigate the effect of different diversification properties. These properties are variable weights and stagnation weights.

Variable weights are first used in the work VW1 and VW2. To improve the diversification capacity, the variables with low flipped frequencies (i.e. low variable weights) are preferred to be selected. It was reported that variable weights improved the diversification capacity of SLS solvers [13].

Stagnation weights are presented in the work [12] as diversification criteria with the purpose of preventing local minima. The stagnation weight of a variable is computed as the frequency of its occurrences in stagnation paths. A stagnation path within a given tenure k is defined as a list of k consecutively flipped variables leading to a local minimum [4].

In order to compute $P_{d_v}(v_i)$, $vw(v_i)$ and $sw(v_i)$ are scaled into $[0, max_{vw} - min_{vw}]$, $[0, max_{sw} - min_{sw}]$ and transferred to $vw'(v_i)$, $sw'(v_i)$ respectively as follows:

$$vw'(v_i) = max_{vw} - vw(v_i) \tag{8}$$

$$sw'(v_i) = max_{sw} - sw(v_i) \tag{9}$$

where $vw(v_i)$ and $sw(v_i)$ are variable weights and stagnation weights respectively. min_{vw} and max_{vw} are the minimum and maximum variable weights. min_{sw} and max_{sw} are the minimum and maximum stagnation weights.

The functions $P_{d_{vw}}(v_i)$ and $P_{d_{sw}}(v_i)$ are the diversification probability distributions of variable weights and stagnation weights respectively. The formulas to compute the scaled diversification criteria $P_{d_{vw}}(v_i)$ and $P_{d_{sw}}(v_i)$ are presented as follows:

$$P_{d_{vw}}(v_i) = \mathcal{N}(vw(v_i)) = \frac{vw'(v_i)}{\sum_j vw'(v_j)} \tag{10}$$

$$P_{d_{sw}}(v_i) = \mathcal{N}(sw(v_i)) = \frac{sw'(v_i)}{\sum_j sw'(v_j)} \tag{11}$$

The probability distributions $P_{d_{vw}}(v_i)$ and $P_{d_{sw}}(v_i)$ grant bigger values to variables with higher scores and low variable weights and low stagnation weights respectively (e.g. preferring the least frequent flipped variables and least stagnated variables).

3.3 Diversification parameter α

According to the probabilistic scoring function (Eq. 5), α is a pre-defined diversification parameter. It specifies the contribution of distribution probabilities in the scoring function in Eq. 5, whereas $(1 - \alpha)$ takes charge of the degree of intensification. In conventional clause-weighting, the objective function $score_w$ is maximized in order to greedily exploit the current searching position. In this case, the degree of intensification is 100% and the diversification parameter α is zero.

4 The PCF algorithm

This section describes our proposed SLS solver, named PCF. It is based on gNovelty$^+$ and employs the new probability-based scoring function. gNovelty$^+$ is the state-of-the-art framework for SLS solvers and some currently superior SLS algorithms for SAT (e.g. Sparrow, CCASat) are tightly correlated with gNovelty$^+$ framework. The adjustment of PCF with the gNovelty$^+$ are listed below:

- Scoring function:
  - Instead of using the $score_w$, PCF applies the new scoring function P described in section 3.2 for both greedy and diversification phases.
  - Re-usage of the additive weighting-scheme for the greediness distribution function P_g
- Diversification criteria:
  - Applying variable weights and stagnation weights as tiebreaks of the scoring function.
  - Variable weights and stagnation weights are contributed in computing the scoring function.

The algorithm PCF is presented in Algorithm 1. PCF utilities a probabilistic objective function to determine search directions. The new heuristics PCF has one extra parameter, the diversification probability α. We use the clause-weighting scoring function in Eq. 2 as the greediness function. According to the diversification criteria of variable weights and stagnation weights respectively, we named the two variants of PCF as PCF$_v$and PCF$_s$.

At the initialization stage, clause weights are initiated to one; variable weights and stagnation weights are set at zero (line 2). If promising variables exist, the promising variable with the maximum probabilistic distribution value P is selected to be flipped, breaking ties by diversification criteria (line 9). Promising variables are defined as the variables whose $score_w$ are positive. If there is no promising variable, the Novelty strategy is invoked to escape from local minima. The procedure of updating stagnation weights is performed according to the original work gNovelty$^+$PCL [12, 5] (line 11). Afterwards, weights of unsatisfied clauses are increased by one according to the additive weighting-scheme in gNovelty$^+$. More specifically, with probability sp, weights of weighted clauses are decreased by one. Weighted clauses are declared as clauses whose weights are larger than one [12]. The variable weight of the selected variable var is increased by one (line 17).

5 Experiments

The experiments were conducted on cbmc [1](a set of software verification problems), crafted instances of the SAT 2012 competition and medium-sized random

[1] http://people.cs.ubc.ca/davet/papers/sat10-dave-instances.zip

Algorithm 1: PCF(Θ, sp)

```
Input  : A formula $\Theta$, $wp = 0.01$, diversification parameter $\alpha$, smooth probability $sp$
Output: Solution $\sigma$ (if found) or TIMEOUT
randomly generate a candidate solution $\sigma$;
initiate all clause weights to 1, stagnation weights and variable weights to 0;
while not timetout do
    if $\sigma$ satisfied the formula $\Theta$ then return $\sigma$ ;
    if within the random walk probability $wp$ then
        $var$ = Random Walk in an unsatisfied clause;
    else
        if there exists promising variable then
            $var$ = variable maximized P function, breaking ties by diversification
            criteria;
        else
            Update stagnation weights;
            $var$ = Novelty Escape with P function, breaking ties by diversification
            criteria;
            Increase the weights of unsatisfied clauses by 1;
            if within smooth probability $sp$ then
                Decrease clause weights of weighted clauses;
    Update candidate solution $\sigma$ with the selected variable $var$;
    Increase the variable weight of $var$ and adapt Novelty noise;
return TIMEOUT;
```

instances of the SAT 2011 competition [2]. In our experiments, the time limit was set at 600 seconds. The number of runs per solver are 50 times for cbmc and 10 times for SAT Competition instances. The two PCF variants in the experiments are PCF$_v$ and PCF$_s$, which employ variable weights and stagnation weights as the diversification criteria. The experiments were conducted on Griffith University Gowonda HPC Cluster Intel(R) Xeon(R) CPU X5650 2.67GHz. Our proposed algorithms PCF$_v$ and PCF$_s$ were compared with seven common SLS solvers:

- VW2, gNovelty$^+$PCL: originally uses variable weights and stagnation weights.
- gNovelty$^+$, sattime2011, EagleUP, Sparrow2011 [2], CCASat: are the top-3 best solvers of SAT competitions.

Table 1. Results on the cbmc, Crafted 2012

Instances	VW2	gNovelty$^+$	sattime2011	EagleUP	Sparrow2011	gNovelty$^+$PCL	CCASat	PCF$_v$	PCF$_s$
cbmc	31%	85%	54%	0%	51%	**100%**	54%	**100%**	**100%**
(39)	439.128	247.997	322.528	600.000	384.359	1.453	276.196	**0.634**	1.013
	-	230,030	354,874	544,647	-	1,287	-	**523**	782
Crafted	0%	88%	84%	23%	70%	88%	82%	93%	**95%**
(74)	600.000	91.896	107.680	464.004	230.463	128.590	115.210	73.552	**70.377**
	-	22,476	22,784	142,440	-	38,067	-	13,725	**13,234**

Table 1 and Table 2 present the results on structured instances (cbmc and crafted 2012) and medium-sized random instances of SAT 2011. Performance of

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

a solver for a specific dataset are reported in three rows. The first and second rows indicate the success rate and the average of median CPU times. The third row specifies the number of flips in thousands. The number of flips of VW2, Sparrow2011 and CCASat are not reported because of the over-flown counted number of flipped in the VW2 , Sparrow2011; and CCASat did not provide that information in the output.

As presented in Table 1, gNovelty$^+$PCL , PCF$_v$ and PCF$_s$ are the three solvers gaining a success rate of 100%. Among them, two variants of PCF performed better than gNovelty$^+$PCL. In regards to crafted instances, the two PCF variants gained better results than other solvers in terms of success rate, CPU time and flips.

Table 2. Results on SAT2011 Medium size

Instances	VW2	gNovelty$^+$	sattime2011	EagleUP	Sparrow2011	gNovelty$^+$PCL	CCASat	PCF$_v$	PCF$_s$
3-SAT	1%	50%	**100%**	**100%**	**100%**	93%	98%	**100%**	**100%**
(100)	595.033	336.962	**0.880**	6.471	20.498	52.893	9.691	1.378	1.966
	-	445, 280	**1, 504**	9, 631	-	98, 343	-	1, 863	2, 688
5-SAT	36%	98%	**100%**	**100%**	**100%**	98%	**100%**	**100%**	**100%**
(50)	410.750	48.136	3.815	30.260	55.846	27.300	7.199	**2.889**	3.277
	-	20, 391	2, 275	14, 925	-	15, 134	-	**1, 280**	1, 473
7-SAT	37%	**100%**	**100%**	**100%**	98%	**100%**	94%	**100%**	**100%**
(51)	400.219	20.026	9.454	23.917	65.350	22.152	51.236	7.944	**7.439**
	-	3, 360	2, 255	5, 301	-	4, 874	-	1, 250	**1, 203**
Random	19%	75%	**100%**	**100%**	100%	96%	98%	**100%**	**100%**
Medium	499.761	184.698	3.785	16.815	40.671	38.726	19.613	**3.420**	3.681
(201)	-	227, 457	1, 887	9, 849	-	53, 928	-	**1, 562**	2, 009

The medium-sized random instances in Table 2 are categorized into three subsets: 3-SAT, 5-SAT and 7-SAT [3]. On random 3-SAT instances, sattime2011, Sparrow2011, PCF$_v$ and PCF$_s$ are the four solvers having a success rate of 100%. Among the four solvers, sattime2011 is the best solver in terms of the average time of 0.88 seconds and the number of flips. Although PCF is not the best solver on 3-SAT instances, the two PCF variants performed well compared with other solvers on 5-SAT and 7-SAT. More specifically, in random 5-SAT, PCF$_v$ is the best solver with an average time of 2.899 seconds. On the other hand, PCF$_s$ performed better than PCF$_v$ with an average time of 7.439 seconds. For the whole dataset, PCF$_v$ is considered better than PCF$_s$ in terms of average CPU time of 3.42 and 3.681 respectively. It is clear from the Table 2, the two PCF variants generally performed well in medium-sized random instances compared with other solvers in the experiments.

Figure 1 illustrates the comparison of solvers for cbmc and SAT 2011 Random Medium and SAT 2012 Crafted instances. The comparison is plotted in the log-log scale cactus presenting the distribution of the number of solved runs when the time limit increases. A run of an instance with a solver is defined as solved if

[3] random k-SAT instances consistently have k variables in every clause

it produces a solution within the given time limit. The x-axis corresponds to the time limit in seconds and the y-axis presents the number of solved runs within the corresponding time limit. The data points in these figures are plotted in every 50 seconds.

According to Figure 1(a) on the cbmc dataset, PCF$_v$, PCF$_s$ and gNovelty$^+$PCL are outperformed other solvers. In the SAT 2011 Random Medium dataset, PCF$_s$, PCF$_v$ and sattime2011 were consistently better than other solvers as displayed in Figure 1(b). The plot displayed in Figure 1(c) indicates that PCF$_s$ and PCF$_v$ steadily improved upon other solvers in crafted instances of the SAT 2012 competition. More specifically, PCF$_s$ was not as good as PCF$_v$ until 150 seconds but surpasses PCF$_v$ thereafter.

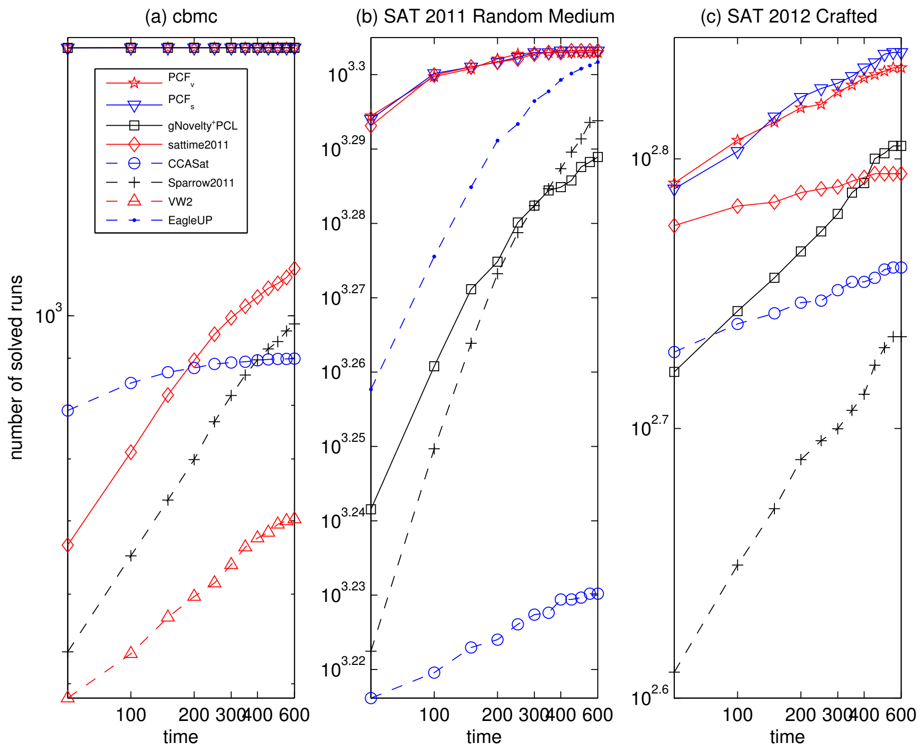

Fig. 1. Log-log scale plot of distribution graph of solved runs over time of cbmc, SAT 2011 medium size random and SAT 2012 Crafted

5.1 Analysis of parameter configurations

The parameter configurations for PCF variants were optimized separately on each instance set within {0.5;1;1.5;2;2.5;3} days by ParamILS with its default setting. ParamILS is a local search optimization tool for parameterized algorithms [9]. The best parameter settings of PCF variants are reported in Table 3. The sets for training and testing were divided half and separately from the original sets.

Table 3. PCF parameters trained by ParamILS

Solvers	cbmc			Crafted			Random		
	tenure	α	sp	tenure	α	sp	tenure	α	sp
PCF$_v$	n/a	0.5	0.1	n/a	0.05	0.30	n/a	0.15	0.30
PCF$_s$	30	0.5	0.1	30	0.05	0.3	15	0.05	0.35

As can be seen in Table 3 for cbmc, the α values are high. In contrast, for crafted and random instances, α converges to low values (i.e. 5% for crafted instances for two PCF variants and 15% and 5% for PCF$_v$ and PCF$_s$ for random instances). The α values for cbmc instances are high to keep the diversification probability high, whereas the diversification probability should be set low for random and crafted instances. This situation probably arises because cbmc instances are highly-constrained problems. In such instances, local minima are results of the conflicts between constraints and the current solution candidate. For this reason, too much greedy exploration in local searching areas will lead search trajectory to trapped areas. This situation, however, is not as problematic as random instances because random instances are formulated by randomization mechanism and are not embedded highly-constrained information in the structures. Therefore, the search trajectory is able to sufficiently overcome the traps and archive the optimal solution.

6 Conclusion and future work

In summary, we proposed a new local search solver for SAT named PCF, which integrates our new probability-based scoring function and variable diversification criteria. The dynamic scoring function is a combination of greediness and diversification capability. The main contribution of this study is modeling the probability distribution for each variable as a dynamic scoring function. An additional contribution of this work is to examine the effect of two diversification criteria of variable properties (e.g variable weights and stagnation weights) in the new scoring function. The probabilistic function is the new regulation between intensification in terms of the clause-weighting score and diversification in terms of variable properties.

The experiments showed that the new solver PCF significantly improved on the performance of other solvers. Comparative experiments demonstrated that the proposed approach outperformed original solvers (e.g. gNovelty$^+$, VW2 and gNovelty$^+$PCL). Furthermore, PCF outperformed other contemporary solvers on structure problems and medium-sized random instances. Our observation from optimization parameters suggests that the diversification parameter α for the verification benchmark and structure instances should be assigned high. In contrast, for random and crafted instances, diversification probability should be defined low.

In future, we plan to apply the probability-based scoring function to other solvers and self-tune the parameters α. Additionally, the intuitive research on the different performance of PCF_v and PCF_s should be investigated thoroughly on different benchmarks.

References

1. Balint, A., Fröhlich, A.: Improving stochastic local search for sat with a new probability distribution. In: SAT. pp. 10–15 (2010)
2. Balint, A., Fröhlich, A., Tompkins, D.A., Hoos, H.H.: Sparrow2011. In: Booklet of SAT-2011 Competition (2011)
3. Cai, S., Su, K.: Configuration checking with aspiration in local search for sat. In: AAAI (2012)
4. Duong, T.T., Pham, D.N., Sattar, A.: A method to avoid duplicative flipping in local search for sat. In: Australasian Conference on Artificial Intelligence. pp. 218–229 (2012)
5. Duong, T.T., Pham, D.N., Sattar, A.: A study of local minimum avoidance heuristics for sat. In: ECAI. pp. 300–305 (2012)
6. Duong, T.T.N., Pham, D.N., Sattar, A., Newton, M.A.H.: Weight-enhanced diversification in stochastic local search for satisfiability. In: IJCAI. pp. 524–530 (2013)
7. Gableske, O., Heule, M.: Eagleup: Solving random 3-sat using sls with unit propagation. In: SAT. pp. 367–368 (2011)
8. Hoos, H.H.: An adaptive noise mechanism for WalkSAT. In: Proceedings of AAAI-02. pp. 635–660 (2002)
9. 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)
10. McAllester, D.A., Selman, B., Kautz, H.A.: Evidence for invariants in local search. In: AAAI/IAAI. pp. 321–326 (1997)
11. Pham, D.N., Duong, T.T., Sattar, A.: Trap avoidance in local search using pseudo-conflict learning. In: AAAI. pp. 542–548 (2012)
12. Pham, D.N., Thornton, J., Gretton, C., Sattar, A.: Combining adaptive and dynamic local search for satisfiability. JSAT 4(2-4), 149–172 (2008)
13. Prestwich, S.: Random walk continuously smoothed variable weights. In: Proceedings of SAT-05. pp. 203–215 (2005)
14. Selman, B., Kautz, H.A.: Domain-independent extensions to gsat: Solving large structured satisfiability problems. In: IJCAI. pp. 290–295 (1993)
15. Selman, B., Levesque, H.J., Mitchell, D.G.: A new method for solving hard satisfiability problems. In: AAAI. pp. 440–446 (1992)
16. Thornton, J.R., Pham, D.N., Bain, S., Ferreira Jr., V.: Additive versus multiplicative clause weighting for SAT. In: Proceedings of AAAI-04. pp. 191–196 (2004)
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.: A switching criterion for intensification and diversification in local search for SAT. JSAT 4(2-4), 219–237 (2008)