VSIDS Branching Heuristics in SAT Solvers and Timed Graph Centrality with Smoothing

Jia Hui Liang[1⋆], Vijay Ganesh[1⋆], and Kaveh Ghasemloo[2]

[1] University of Waterloo
[2] University of Toronto

Abstract. State of the art SAT solvers crucially rely on Conflict-Driven Clause-Learning (CDCL) and Variable State Independent Decaying Sum (VSIDS) branching heuristic for their performance. In particular, VSIDS is the best branching technique known to this date. Many experiments attest to the veracity of this observation. However, despite the widespread use of VSIDS, it is still not well-understood.

In this paper, we advance our understanding of VSIDS by showing a strong relationship between VSIDS and *exponentially smoothed timed or temporal graph centrality* (TGC) measures. More precisely, VSIDS viewed as a ranking of variables *very strongly correlates* with the ranking produced by TGC measures over the *timed or temporal variable incidence graphs* (TVIG) of original and learnt clauses, throughout the run of the CDCL solver over the benchmarks we considered. In addition, the variables selected by VSIDS for branching have high TGC measures. To come to these empirical conclusions, we performed a total of 31 experiments with 35 million ranking measurements using the Sat4j SAT solver over a benchmark of 300 Application instances from the SAT Competition 2013.

VSIDS has two main components: an additive bumping of variable activities, and a multiplicative decay of activities of all variables. Our results indicate that the additive bumping of a variable corresponds to increasing the graph centrality measure of the corresponding vertex in TVIG. On the other hand, the multiplicative decay corresponds to exponential smoothing aspect of the TGC measures. These two components of VSIDS combined together result in the selection of branching variables with high timed degree centrality (TDC) and timed eigenvector centrality (TEC) measures in the TVIG of the original and learnt clauses, as the TVIG evolves through the solving process.

1 Introduction

The Boolean satisfiability (SAT) problem [12] is the quintessential NP-complete problem [12, 34], a class of decision problems widely conjectured to be hard to solve efficiently. Yet, impressively, modern sequential Conflict-Driven Clause-Learning SAT solvers [13, 9, 27, 28, 5] are widely used to solve large industrial

⋆ Joint first authors

instances [21, 2]. Although hundreds of techniques and heuristics have been proposed to improve their performance [1, 2], modern SAT solvers rely crucially only on a handful of them. Of these, the two most important are Conflict-Driven Clause-Learning with backjumping [28] (CDCL) and Variable State Independent Decaying Sum (VSIDS) branching heuristic [30]. Many systematic experiments have been performed to ascertain the veracity of this observation [21]. Additionally, not only is VSIDS one of the best branching heuristics, but all other high-performing branching heuristics are simply variants of VSIDS.

Researchers have already proposed some theoretical explanations for the impact of clause-learning on the performance of the modern SAT solvers: clause-learning allows SAT solvers to polynomially simulate *general* resolution propositional proof system [32, 4, 6]. However, our understanding of the role played by VSIDS heuristic is limited. It is indeed a very curious phenomenon that VSIDS and its variants play such a significant role in the performance of SAT solvers.

The motivation for the research presented in this paper is to achieve a better understanding of VSIDS. We hope that an improved understanding of VSIDS will lead to more scalable and faster SAT solvers. It can also help in the classification of the instances where these SAT solvers perform well, and formally explaining their performance on those instances. In the following, by the term VSIDS we refer to a family of branching heuristics, all of which share a core set of features, namely, additive bumping and multiplicative decay. In particular, we implemented Chaff's [30], MiniSAT's [13], and Sat4j's [25] variations of VSIDS into Sat4j and studied their behavior.

Our Findings: In the current paper, we take steps towards a better understanding of VSIDS on industrial instances by identifying certain key properties of VSIDS and the variables selected by it. We start by observing that VSIDS, when viewed as a ranking function, has the property that at any point throughout the solving process the *activities* of variables maintained by it induces a ranking on variables in the SAT formula input to a CDCL SAT solver. Starting from this simple observation we made the following findings:

First, we empirically establish that these VSIDS rankings correlate very strongly with the variable rankings induced by *exponentially smoothed timed or temporal graph centrality* (TGC) measures over the *timed or temporal variable incidence graphs* (TVIG) of original and learnt clauses of a SAT instance input to a CDCL solver. TVIG is an extension of the well-known *variable incidence graph* (VIG) of CNF formulas and incorporates the dynamically evolving aspect of the learnt clause database inside SAT solver and uses exponential smoothing to focus on recently learnt clauses. TGC is the temporal version of the widely-used graph centrality measures which are used to identify important and central vertices in a graph. The definitions are inspired by recent research on temporal aspects of social networks [17, 24, 33, 36]. For example, the timed PageRank algorithm [36] is used to discover important publications that are likely to be referenced in the future based on the extant citation graph.

Second, we show that VSIDS without decay (i.e., only additive bumping) correlates strongly with non-timed graph centrality (NGC) rankings over the

VIG. Finally, we show that VSIDS typically selects variables with high timed degree centrality (TDC) and timed eigenvector graph centrality (TEC) measures. Section 2 provides detailed definitions of the concept referred to here.

Putting the above-mentioned findings together, we have a single family of graph-theoretic measure, namely, TGC that succintly characterizes both the additive bump and multiplicative decay components of VSIDS family of heuristics. Variables that have high centrality correspond to variables in "recent" learnt clauses and get additively bumped, whereas variables that are exponentially decayed according to TGC correspond to variables that do not occur "robustly" in recent learnt clauses and are multiplicatively decayed. Also, through our experiments we note that the top-ranked unassigned VSIDS variable have high TGC rankings.

Brief Overview of Experiments Performed: In order to come to these empirical conclusions, we conducted 31 experiments, performed over 35 million ranking comparisons over a benchmark suite of 300 Boolean formulas from the Applications category of the SAT Competition 2013 [2]. For almost all the Boolean formulas we considered, except a couple, we found that the mean of the correlations between VSIDS and TGC rankings is very strong. To be more precise, the standard deviation for the means we computed in these experiments was small. The details of can be found in Section 4.

Contributions: We make the following contributions in this paper. In the following, VSIDS is viewed as a ranking function:

1. VSIDS rankings with decay correlate very strongly with TGC rankings, according to the standard Pearson correlation coefficient [31].
2. VSIDS rankings without decay correlate strongly with NGC rankings, according to the standard Pearson correlation coefficient [31].
3. VSIDS with decay typically selects variables with high TDC and TEC.
4. VSIDS without decay typically selects variables with high non-timed degree centrality (NDC) and non-timed eigenvector centrality (NEC).
5. We present an analytical argument for the correspondence between the original Chaff VSIDS and TDC with exponential smoothing, under certain assumptions.
6. Our results indicate that the bumping component of VSIDS corresponds to increasing graph centrality of bumped variables, while the decay component of VSIDS corresponds to exponential smoothing of temporal aspect clause learning during the solving process.
7. All experimental data and appropriate code is available from our website: `https://ece.uwaterloo.ca/~vganesh/vsids-centrality.html`

2 Preliminaries

In this section we provide the necessary background and definitions for later sections of the paper. We start with a brief overview of the VSIDS family of branching heuristics.

Throughout the paper F denotes an input CNF formula to a CDCL SAT solver using VSIDS; Var is the set of variables in F, often identified with the set of its clauses; Γ denotes the set of (original and learnt) clauses in the solver's clause database; c denotes a clause and $|c|$ denotes its length (the number of literals in the clause). V denotes the set of vertices in a graph; E denotes the set of edges in a graph; and $w(e)$ is the weight of edge e.

2.1 CDCL SAT Solvers and the VSIDS Branching Heuristic

The VSIDS (Variable State Independent Decaying Sum) branching heuristic was first proposed by the authors of the Chaff SAT solver [30]. Many variants of VSIDS have since been proposed, and it is a common practice in the literature to refer to any of these variants simply as VSIDS. We focus on three of the more well-known variants of VSIDS: cVSIDS refers to the Chaff or zChaff version [27] of VSIDS, mVSIDS refers to the MiniSAT version 1.14 [13] variant, and sVSIDS refers to the Sat4j version 2.3.5 [25] variant. All three have the following basic common characteristics:

1. **Activity score:** VSIDS assigns to each variable v in Var a floating-point number called *activity* score of v which we denote by $act(v)$.
2. **VSIDS ranking:** Consider the ranking of variables in Var according to their activity scores in the decreasing order. The solver uses this ranking to decide the next branching variable. Typically the highest-ranked unassigned variable is selected for branching.
3. **Initialization:** At the start of the solving process, the activity scores of all variables are initialized (usually to 0).
4. **Additive bump:** When the solver learns a clause, a subset of variables in Var is chosen, and the activities of these variables are additively increased. The amount of the increase is called the (additive) *bump*. This value is typically set to 1.
5. **Multiplicative decay:** Once in a while the activities of all variables are multiplied by a constant $0 < \alpha < 1$ called the (multiplicative) *decay factor*.

We next look at the the differences between the three VSIDS variants:

1. **cVSIDS:** The activities of variables occurring in the learnt clause are bumped up by 1. The activities of all variables in Var are multiplied by a constant $0 < \alpha < 1$. The decay occurs after every i conflicts. The original Chaff implementation set $i = 256$. We follow the policy used in more recent solvers like MiniSAT and use $i = 1$.
2. **mVSIDS:** The activities of all variables participating in the conflict analysis leading to the learnt clause (including the variables in the learnt clause) are bumped up by 1 once. The activities of all variables are decayed in the same manner as in cVSIDS.
3. **sVSIDS:** The activities of all variable participating in the conflict analysis leading to the learnt clause (including the variables in the learnt clause) are bumped up by 1 as many times as their incoming degree in the *conflict*

analysis graph (CAG, the part of the implication graph participating in conflict analysis). The activities of all variables are decayed in the same manner as in cVSIDS.

Next, we go over a common presentation of CNF formulas as graphs.

2.2 Variable Incidence Graph (VIG)

The VIG of the CNF formula F is defined as follows: Vertices of the graph are the variables in the formula. For every clause $c \in F$ we have an edge between each pair of variables in c. In other words, each clause corresponds to a clique between its variables. The weight of an edge is $\frac{1}{|c|-1}$ where $|c|$ is the length of the clause. Each clause increases the in-degree of its variables by 1. VIG does not distinguish between positive and negative occurrences of variables. We combine all edges between each pair of vertices into one weighted edge, e.g. if variables x and y occur together in 7 clauses of size 3 then the edge between x and y has weight $\frac{7}{3-1} = \frac{7}{2}$.

Definition 1 (VIG). *The VIG of a CNF formula F is a weighted graph defined as follows: set of vertices $V = Var$, set of edges $E = \{xy \mid x, y \in c \in \Gamma\}$, and the weight function $w(xy) = \sum_{x,y \in c \in \Gamma} \frac{1}{|c|-1}$.*

The VIG representation of CNF formulas is used by the solvers for their conflict analysis graphs, and the cVSIDS, sVSIDS and mVSIDS heuristics implicitly use this representation to compute which variables to additively bump up.

The VIG ignores the temporal aspect of learnt clauses. To incorporate the temporal aspect of learnt clauses we introduce *temporal variable incidence graph*.

2.3 Timed Variable Incidence Graph (TVIG)

The full temporal variable incidence graph (FTVIG) of a solver on input formula F is a labeled weighted multi-graph defined as follows: the vertices are the variables in F, for each pair of variables there is an edge between them for each (original or learnt) clause that they occur together. The weight of an edge is $\frac{1}{|c|-1}$ where $|c|$ is the length of the clause. In addition each edge is labeled with a timestamp denoted by $t(e)$. The timestamp of an edge from an original clause is 0. The timestamp of an edge from a learnt clause is the number of conflicts up to the learning of the corresponding clause. We refer to the difference between the current time t and the timestamp of an edge $t(e)$ as the age of a edge: $age(e) = t - t(e)$.

Fix an *exponential smoothing factor* $0 < \alpha < 1$. The (aggregated) *temporal variable incidence graph* (TVIG) at time t is a weighted graph obtained from the FTVIG as follows: we ignoring the edges with timestamp larger than t and combining the remaining edges between each pair of vertices x and y into a single edge between them. The weight of the resulting edge is $\sum_{e \in E(x,y)} w(e)\alpha^{age(e)}$.

Consider the changing TVIG throughout the solving process: as new clauses are learnt new edges are added to the graph, and the time increases. The weight of an edge in TVIG decreases exponentially with time if no new learnt clause contains its variables.

Definition 2 (TVIG). *The TVIG of a clause database at time t is the defined in the same way as VIG except with a modified weight function that takes the ages of edges into:* $w(xy) = \sum_{x,y \in c \in \Gamma} \frac{\alpha^{age(c)}}{|c|-1}$

Next, we overview the non-timed graph centrality measures (NGC) and timed graph centrality measures (TGC).

2.4 Degree Centrality (NDC) and Eigenvector Centrality (NEC)

A graph centrality measures is function that assigns a real number to each vertex in a graph. The number associated with each vertex denotes its relative importance in the graph [16, 15, 35]. For example, according to the degree centrality [15] measure, a vertex with the most neighbors is deemed the most important. Centrality measures are widely used models for understanding ranking and recommendation functions in Internet search and social networks, e.g. Google PageRank[10], HITS[23], and SALSA [26].

Definition 3 (NDC). *The NDC of a vertex is defined as the degree of the vertex. The NDC ranking is obtained by sorting nodes in decreasing order according to their weighted incoming degree in the VIG.*

Degree centrality considers a vertex more important if it has more neighbors. Degree centrality is a very local measure, e.g. the importance of a vertex does not depend on the importance of its neighbors. Eigenvector centrality is a natural extension of degree centrality to overcome its locality and account for the propagation of importance. It is the de facto standard for determining the importance of vertices in social networks [20].

Definition 4 (NEC). *The NEC of a vertex is defined as its corresponding value in the eigenvector of the greatest eigenvalue of the graph's adjacency matrix. We refer to this eigenvector as the dominant eigenvector. The NEC ranking is obtained by sorting nodes in decreasing order according to their respective values in the dominant eigenvector of the adjacency matrix of the VIG.*

Note that the components of the dominant eigenvector are all non-negative. Assume that each vertex's importance flows to its neighbors and each of them receives a fraction of the importance proportional to the weights of the edges between them. Eigenvector centrality assigns to each vertex a non-negative amount of importance such that the importance flowing into a vertex is equal to the importance flowing out of it.

Degree centrality can be consider as an approximation of eigenvector centrality in the following sense. A standard algorithm to compute the eigenvector

centrality is the power iteration method [18]. In this algorithm we take the adjacency matrix A of a graph, iteratively multiply the adjacency matrix with an initial column vector of all ones $\mathbf{1}$: $\mathbf{1}, A\mathbf{1}, A^2\mathbf{1}, \ldots, A^i\mathbf{1}, \ldots$ This sequence eventually converges to the dominant eigenvector of A, under appropriate normalization of the column vector at every iteration. In practice, the algorithm is stopped when the error becomes small enough, e.g. after 100 iterations. The output of the first iteration, $A\mathbf{1}$, is the degree centrality.

These NGC measures view all edges similarly, there is no distinction between edges of an original clause and the edges of a recently learnt clause. To incorporate the temporal aspect of edges we introduce *temporal degree and eigenvector graph centrality* measures with *exponential smoothing* (TGC).

2.5 Temporal Degree Centrality (TDC) and Temporal Eigenvector Centrality (TEC)

NGC measures discussed in the previous section are static measures of importance of vertexes in a graph. By contrast, consider a graph that changes over time, e.g. the VIG of the initial and learnt clauses throughout the solving process. The temporal nature of these changes might contain useful information that the static centrality measures will ignore. It is often the case that more recent data points are more useful than older data points. Using this idea we penalize the weights of the older edges. A natural way to penalize the older edges is exponentially reducing the weight of edges by their age, resulting in what is called *exponential moving average* [11].

The key observation is to incorporate this temporal information inside the graph as we discussed in section 2.3. More generally, let H be a weighted multigraph where each edge e in H has a positive number as its label which we will refer to as $age(e)$. The (aggregate) temporal graph and temporal graph centrality measures with exponential smoothing are defined as follows. Fix a exponential smoothing factor $0 < \alpha < 1$. We first turn this labeled weighted muli-graph into an unlabeled weighted graph G. For each pair of vertices x and y let $E(x, y)$ denote the set of edges in H between x and y. To obtain G we remove edges in $E(x, y)$ and replace them with a single edge of weight $\sum_{e \in E(x,y)} w(e)\alpha^{age(e)}$. TDC and TEC measures of H are defined as degree and eigenvector graph centrality measures of the aggregate graph G. Note that FTVIG and TVIG are a special case of these definitions.

Definition 5 (TDC). *The TDC of a clause database at time t is defined as the degree centrality of TVIG at time t. The TDC ranking is obtained by sorting nodes in decreasing order according to their weighted incoming degree in the TVIG.*

Definition 6 (TEC). *The TEC of a clause database at time t is defined as the eigenvector centrality of TVIG at time t. The TEC ranking is obtained by sorting nodes in decreasing order according to their eigenvector centrality in the TVIG.*

3 cVSIDS Centrality Theorem

In this Section, we informally introduce the theorem that states that cVSIDS rankings matches TDC rankings under certain conditions. The formal theorem and proof are in the Appendix.

Informally, the theorem says that the cVSIDS and TDC ranking always match exactly assuming: no clause deletion/in-processing (i.e., clauses are never removed or altered), and the solver seeds the initial activity scores so that the rankings start off equal for the original input formula. It is easy to prove this theorem through induction over time steps during the run of the solver, i.e., if at any point in time $t = i$ the two rankings match then they will remain matched for all times in the future. To see this, observe that the way TDC rankings of all variables in a clause added at time step $t = i + 1$ are bumped by 1, just like in cVSIDS. Both cVSIDS and TDC implement the same exponential smoothing. Hence, if the ranking matched at time step $t = i$, then they ought to remain matched into the future.

4 Experiments and Results

In this section, we describe the experimental setup, methodology, and results to support the following hypotheses made in this paper, namely, that

1. **VSIDS vs. TGC (6 experiments):** The VSIDS variants cVSIDS, mVSIDS, and sVSIDS (with decay), and viewed as ranking functions, correlate very strongly with both TDC and TEC according to Pearson correlation coefficient and top-10 measure.
2. **VSIDS vs. TGC with no clause deletion and no in-processing (6 experiments):** The preceding claim is true even when clause deletion and in-processing are switched off. In other words, we get the same results when controlling for clause deletion and in-processing.
3. **VSIDS vs. NGC (6 experiments):** The VSIDS variants cVSIDS, mVSIDS, and sVSIDS (with decay), and viewed as ranking functions, do not correlate well with either NDC or NEC according to Pearson correlation coefficient and top-10 measure.
4. **VSIDS without decay vs. NGC (6 experiments):** The VSIDS variants cVSIDS, mVSIDS, and sVSIDS without decay, and viewed as ranking functions, correlate well with both NDC and NEC according to Pearson correlation coefficient and top-10 measure.
5. **VSIDS without decay vs. TGC (6 experiments):** The VSIDS variants cVSIDS, mVSIDS, and sVSIDS without decay, and viewed as ranking functions, do not correlate as well as the previous set of experiments with either TDC or TEC according to Pearson correlation coefficient and top-10 measure.

In addition, we also experimentally verified that the cVSIDS Centrality Theorem holds under appropriate assumptions. The theorem along with the appropriate experiments are detailed in the Appendix. Note that this experiment is in addition to showing that cVSIDS ranking correlates very strongly with TDC even in the presence of clause deletion/in-processing and no special seeding of the variable activities.

4.1 Experimental Setup

All the experiments were performed with Sat4j version 2.3.5 [25] on all 300 CNF Boolean formulas obtained from the Application category of the SAT Competition 2013 [2]. The reason we chose Sat4j is because it implements a minimal set of features required for a full-fledged CDCL solver, and the absence of extra features simplifies our analysis allowing us to isolate the effects of the VSIDS carefully. In addition, Sat4j's modular design is ideal for modifying and experimenting.

Sat4j implements CDCL with one main loop. Every decision and conflict is one iteration of this loop. We take measurements on the current state of the solver after every 1000 iterations. The search terminates once the solver finds a satisfiable solution or is unsatisfiable or times out after 5000 seconds. Each instance is given 5GB of JVM heap space. At the end of the run, we record the mean Pearson correlation coefficient [31], and the percent of top-10 measure for each instance.

We use 100 iterations of the power iteration algorithm to compute NEC/TEC, and 1 iteration for NDC/TDC. Popular graph libraries such as NetworkX and SNAP use 100 iterations by default to compute the eigenvector centrality of a graph. We follow their example. All experiments were performed on the SHARCNET cloud [3], where cores range in specs between 2.2 to 2.7 GHz. SHARCNET provides high performance computing (HPC) infrastructure for running computationally intensive experiments in parallel.

4.2 Experimental Methodology

The overall experimental methodology adopted here is based on comparing the rankings produced by the VSIDS family of heuristics and the GC family of measures. Observe that we compare the rankings dynamically as the solver solves an instance, and not just once at the begining of the solver's run. Such a dynamic comparison gives us a much better picture of the correlation between two different ranking functions or measures than a single point of comparison.

4.2.1 The Measurements

For each experiment, let V represent the chosen VSIDS variant and T represent the chosen GC variant. We implemented both V and T in Sat4j. We then ran Sat4j on the 300 different CNF Boolean formulas obtained from the Applications category of the SAT 2013 competition. For each run, at an interval of every 1000 iterations, we dump the VSIDS ranking (and the activity scores) of all the variables in the input formula according to the heuristic V. At these very same intervals, we compute the GC ranking according to T over the VIG/TVIG

of the original input and learnt clauses. We perform, on average, around 3763 such measurements per run. We made a total of over 35 million measurements in the 31 experiments.

4.2.2 Methodology for Comparing Rankings based on Pearson Correlation Coefficient

For each set of experiments, for each SAT instance, for every measurement made, we compute the Pearson product-moment correlation coefficient [31] between the variables' VSIDS activities and their centrality. This is the most commonly used correlation coefficient in statistics for measuring the degree of linear relationship between two sets of rankings or data. The Pearson correlation coefficient is conventionally interpreted as follows [14]: 0.00–0.19 is a very weak correlation, 0.20–0.39 is a weak correlation, 0.40–0.59 is a moderate correlation, 0.60–0.79 is a strong correlation, 0.80–1.00 is a very strong correlation. We compute the average of the Pearson correlation coefficient over the execution of a SAT solver on each instance. We chose the Pearson correlation coefficient because it is the standard measurement for correlation in the literature. We use this measurement to understand the degree of similarity between VSIDS and centrality rankings.

4.2.3 Methodology for Comparing Rankings based on the Top-10 Measure

Another very useful metric to compare the rankings is what we call the top-10 measure. The idea is to record how often the top-ranked unassigned variable according to VSIDS occurs in the top-10 ranked variables according the TGC variant we are comparing against. If the VSIDS top-ranked unassigned variable occurs very often among the top-10 ranked variables according to GC, then we infer that VSIDS picks variables that are highly ranked according to GC. The rationale for this metric is that SAT solvers typically only choose the top-ranked unassigned variable, according to the VSIDS ranking, to branch on.

Let U be the current list of unassigned variables and let v be the highest ranked variable among them according to some VSIDS variant. Let i be the position of variable v according to a specific GC ranking, excluding assigned variables. Then the top-10 measure is defined as follows: The *top-k measure* is 1 if $i \leq k$, otherwise 0.

In our experiments we set $k = 10$. Again, we compute the average of top-10 measure over the execution of a SAT solver on each instance. We decided to include this measure to better understand the variables VSIDS picks to branch on, hence it only focuses on the unassigned variable with the highest activity. We use the top-10 measure to show that VSIDS branches on highly central variables.

4.2.4 The Reporting of Results

For every pair of rankings, one from the VSIDS family and the other from GC family, we report the following numbers: We compute the top-10 measure and Pearson correlation coefficient between the pair of rankings every 1000 iterations. On termination, we compute the average for the instance. We take all

the instance averages and average them again, and report the average the averages. These numbers are labeled as "mean top-10" or "mean Pearson" in the tables below. For example, in the tables below, a "mean top-10" of 0.904 is interpreted as "for the average instance in the experiment, 90.4% of the measured top ranked variables according to VSIDS are among the 10 unassigned variables with the highest centrality". A high "mean top-10" implies VSIDS is picking highly central decision variables most of the time in that experiment. Likewise, a high "mean Pearson" implies the average instance has a strong positive linear correlation between VSIDS activity score and GC.

4.3 VSIDS vs TGC

	cVSIDS vs TDC	mVSIDS vs TDC	sVSIDS vs TDC
Mean top-10	0.904	0.763	0.655
Mean Pearson	0.970	0.847	0.862

	cVSIDS vs TEC	mVSIDS vs TEC	sVSIDS vs TEC
Mean top-10	0.636	0.625	0.557
Mean Pearson	0.843	0.736	0.753

Table 1. Results of VSIDS vs TGC.

In this Subsection, we present two results: In the first (resp. second) table above, we compare all three variants of VSIDS (with multiplicative decay) vs. TDC (resp. TEC). As is evident from the data, VSIDS is very strongly correlated with TDC, in particular, the 0.970 Pearson correlation between cVSIDS and TDC is exceptionally high. The metrics are lower with TEC, but the correlation is still strong.

4.4 Experiment: VSIDS (no clause deletion, no in-processing) vs TGC

	cVSIDS vs TDC	mVSIDS vs TDC	sVSIDS vs TDC
Mean top-10	0.924	0.776	0.661
Mean Pearson	0.980	0.855	0.865

	cVSIDS vs TEC	mVSIDS vs TEC	sVSIDS vs TEC
Mean top-10	0.650	0.639	0.566
Mean Pearson	0.855	0.749	0.764

Table 2. Results of VSIDS with no clause deletion and no in-processing vs TGC.

This experiment is similar to the first experiment except clause deletion and in-processing are switched off. We see minor improvements in the results. This shows that our hypotheses holds after controlling some of the effects external to VSIDS, namely clause deletion and in-processing.

4.5 Experiment: VSIDS vs NGC

	cVSIDS vs NDC	mVSIDS vs NDC	sVSIDS vs NDC
Mean top-10	0.179	0.221	0.222
Mean Pearson	0.412	0.393	0.431

	cVSIDS vs NEC	mVSIDS vs NEC	sVSIDS vs NEC
Mean top-10	0.182	0.230	0.232
Mean Pearson	0.389	0.349	0.385

Table 3. Results of VSIDS vs NGC.

This experiment is similar to the first one except NGC is used in place of the TGC. The top-10 and Pearson correlation dropped significantly compared to the first experiment. The Pearson correlation borders between weak and moderate. The low top-10 measure suggests that VSIDS is no longer focused on highly central variables when centrality is defined without time. This supports our notion that time is integral in graph centrality to properly understand the behavior of VSIDS.

4.6 Experiment: VSIDS without Decay vs NGC

	cVSIDS vs NDC	mVSIDS vs NDC	sVSIDS vs NDC
Mean top-10	0.455	0.446	0.474
Mean Pearson	0.729	0.628	0.733

	cVSIDS vs NEC	mVSIDS vs NEC	sVSIDS vs NEC
Mean top-10	0.430	0.480	0.450
Mean Pearson	0.723	0.664	0.704

Table 4. Results of VSIDS without decay vs NGC.

In this experiment, we compare VSIDS without decay (i.e. only bumping) with NGC. The Pearson correlation is strong but weaker than the first experiment, as expected.

4.7 Experiment: VSIDS without Decay vs TGC

	cVSIDS vs TDC	mVSIDS vs TDC	sVSIDS vs TDC
Mean top-10	0.406	0.424	0.404
Mean Pearson	0.654	0.578	0.588

	cVSIDS vs TEC	mVSIDS vs TEC	sVSIDS vs TEC
Mean top-10	0.384	0.408	0.393
Mean Pearson	0.606	0.526	0.525

Table 5. Results of VSIDS without decay vs TGC.

This experiment is similar to the preceeding one except it is conducted with TGC. The Pearson correlation and top-10 measure is noticeably weaker, as expected.

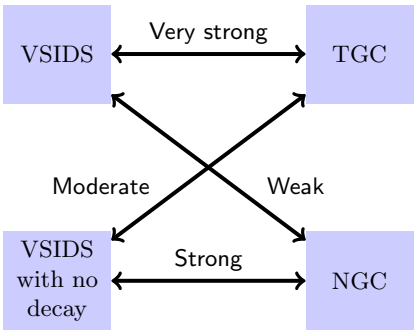

Fig. 1. Summary of the Pearson corrleation between VSIDS and centrality.

5 Interpretation of Results

We started with the hypotheses that the ranking produced by the VSIDS family of branching heuristics strongly correlates with the TGC family of graph measures, and the correlation remains strong during the entire run of a CDCL SAT solver as the TVIG of the learnt+original clauses evolve. A second set of hypotheses we posited is that the VSIDS family of branching heuristics without decay strongly correlates with the NGC family of graph measures. A third hypothesis was that the highest ranked unassigned variable according to VSIDS would also have high TGC, and again the correlation would remain strong during the entire run of a CDCL solver. We designed careful experiments to verify these hypotheses, and found them to be true.

In the VSIDS with decay vs. TGC experiments (Subsection 4.3), we used the Pearson correlation coefficient to show that the two rankings are *very strongly correlated*. Given that clause deletion and in-processing can significantly influence the TVIG, we controlled for this effect in our second set of experiments (Subsection 4.4). We found that the correlation is only strengthened when we switch-off clause deletion and in-processing. An obvious next question that the reader might ask is "Do we really need timed versions of graph centrality?" So we performed another set of experiments (Subsection 4.5) where we compared VSIDS with decay with NGC measures. As expected, the correlation was weak. In the VSIDS without decay vs. NGC experiments (Subsection 4.6), we used the Pearson correlation coefficient to show that the additive bump part of the VSIDS variants *strongly correlate* with graph centrality measures (without time). Finally, by comparing VSIDS variants without decay (only additive bump) with TGC measures we found that these two rankings were moderately correlated (Subsection 4.7).

A natural question the reader might have is "what does all this mean?" Alternatively, how does the lens of centrality measures help us better understand the additive and multiplicative decay components of VSIDS. We can make this question more precise by asking the following two questions: First, what is the effect of additive bumping in terms of centrality of variables (nodes) in TVIG?

Second, What is the effect of multiplicative decay on VSIDS ranking or centrality of variables in the TVIG?

From our experiments, at least for the solver and benchmarks considered, we can say that for all the VSIDS variants considered in this paper additive bumping increases the centrality of the chosen variables (e.g., the variables in the most "recent" learnt clause in cVSIDS). We also observe from our results that the top-ranked variable that solvers pick for branching have very high TGC rank. It could be that branching on central nodes crucially influences solver performance.

The answer to "why perform multiplicative decay" is bit more involved. We confirm the interpretation first proposed by the work by Moskwicz et al. [30] in their patent application [29], later confirmed by Armin Biere [8] and Huang et al. [19]. In essence, the traditional view is that the multiplicative decay over time acts as an exponential smoothing average that is biased towards ranking or weighing highly those variables that consistently occur in "recent" conflicts, and decaying away the ranks of all other variables. Based on our results, a complimentary view is that multiplicative decay acts as an exponential smoothing average that is biased towards ranking highly those variables that consistently remain central over time as the TVIG evolves, and decaying away the ranks of all other variables.

6 Related Work

The possibility of a connection between eigenvector centrality and branching heuristics was first proposed by Katsirelos and Simon [22]. In their paper, the authors computed eigenvector centrality (using the specific definition provided by Google PageRank) only once on the original input clauses and showed that most of the decision variables have higher than average centrality. Also, it bears stressing that their definition of centrality is not temporal.

By contrast, our results correlate the ranking due to variants of VSIDS very strongly with variants of TGC, and importantly, dynamically over the run of the solver. In addition, we proved and empirically verified the cVSIDS Centrality Theorem that establishes an exact correspondence between cVSIDS and TDC under the assumption that the activity scores of all variables in the input formulas are seeded by their respective degree centralities and no clause-deletion and no in-processing. Even in the presence of clause-deletion/in-processing and no special seeding of activities, the correlation between cVSIDS and TDC is very strong based on the Pearson's ranking correlation. In addition, we show that the ranking due to additive bump component of VSIDS correlates strongly with rankings based on NGC measures.

While our empirical results are based on the most recent benchmark suite from the SAT Competition 2013 (the entire Applications category) using a single solver Sat4j, we believe that it will carry over for other solvers and benchmarks. The reason we believe this is that VSIDS implementations tend to be very similar across different solvers.

6.1 A Brief History of VSIDS

VSIDS was a major breakthrough when first introduced as part of the Chaff solver by Moskewicz et al. The key idea is to collect statistics from conflict clauses to guide the direction of the search, where recent conflict clauses are favored. Another positive characteristic of the heuristic is the low computational overhead. Moskewicz et al explicitly called their technique a low-pass filter in their paper [30] and patent [29]. Huang et al [19] describe VSIDS even more precisely as an exponential moving average (i.e. exponential smoothing, a form of low-pass filter) over the frequency of variable occurences in "recent" learnt clauses. In 2008, Biere [7] reformulates VSIDS such that the activities are rational numbers between 0 and 1, and 'can be interpreted as the percentage of the number of times a variable was involved in a conflict "recently"'. In 2009, Biere [8] reaffirmed VSIDS as a low-pass filter. MiniSAT [13] introduced an improved variant by bumping all the variables in the conflict analysis. The frequency of the decay was increased to every conflict to better respond to changes in the recent conflicts. Sat4j [25] made a modification to bump proportionally to the indegree of the conflict analysis graph.

6.2 How our Work is Different

We provide an alternate complementary characterization to the low-pass filter picture originally painted by the authors [30] of the VSIDS heuristic. More precisely, we show that the top-ranked VSIDS variables also have high centrality according to the TGC rankings, based on Pearson's correlation and top-10 measure. This alternate characterization can be very valuable in eventually understanding why VSIDS is effective, and open up research for a plethora of new branching heuristics.

7 Conclusions

In this paper we provide three results. First, we showed that for the solver and benchmarks we considered, the rankings produced by the VSIDS variants we considered, namely, cVSIDS, mVSIDS, and sVSIDS, correlated very strongly with rankings produced by TGC measures over the TVIG of the input and learnt clauses of a SAT instance input to a solver. Second, we showed that the ranking due to additive bump component alone (VSIDS without decay) correlates strongly with the corresponding GC rankings. Note that these correlation remains strong throughout the run of the solver. Finally, we showed that the top-ranked unassigned variable according to the VSIDS ranking also has high TGC measure. These results put together show that graph centrality is an important lens through which to understand the behavior of the VSIDS family of branching heuristics.

References

[1] Proceedings of Past SAT Conferences, 2013. `http://www.satisfiability.org`.

[2] SAT Competition Website, 2013. `http://www.satcompetition.org`.

[3] SHARCNET Website, 2013. `https://www.sharcnet.ca`.

[4] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. In *Theory and Applications of Satisfiability Testing-SAT 2009*, pages 114–127. Springer, 2009.

[5] Gilles Audemard and Laurent Simon. Glucose: a solver that predicts learnt clauses quality. *IJCAI*, 9:399–404, 2009.

[6] Paul Beame, Henry A Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.(JAIR)*, 22:319–351, 2004.

[7] Armin Biere. Adaptive restart strategies for conflict driven sat solvers. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing*, SAT'08, pages 28–33, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] Armin Biere. P {re, i} cosat@ sc09. *SAT*, 4:41–43, 2009.

[9] Armin Biere. Lingeling, 2010.

[10] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[11] Robert G Brown. *Exponential Smoothing for predicting demand*. Little, 1956.

[12] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[13] Niklas Een and Niklas Sörensson. MiniSat: A SAT solver with conflict-clause minimization. *Sat*, 5, 2005.

[14] James D Evans. *Straightforward statistics for the behavioral sciences*. Brooks/Cole Publishing Company, 1996.

[15] Katherine Faust. Centrality in affiliation networks. *Social Networks*, 19(2):157 – 191, 1997.

[16] L Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

[17] P.A. Gloor, J. Krauss, S. Nann, Kai Fischbach, and D. Schoder. Web science 2.0: Identifying trends through semantic social network analysis. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 4, pages 215–222, Aug 2009.

[18] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[19] Ruoyun Huang, Yixin Chen, and Weixiong Zhang. Sas+ planning as satisfiability. *J. Artif. Int. Res.*, 43(1):293–328, January 2012.

[20] Muhammad Usman Ilyas and Hayder Radha. Identifying influential nodes in online social networks using principal component centrality. In *ICC*, pages 1–5. IEEE, 2011.

[21] Hadi Katebi, Karem A. Sakallah, and João P. Marques-Silva. Empirical study of the anatomy of modern sat solvers. In *Proceedings of the 14th International Conference on Theory and Application of Satisfiability Testing*, SAT'11, pages 343–356, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] George Katsirelos and Laurent Simon. Eigenvector centrality in industrial sat instances. In Michela Milano, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 348–356. Springer Berlin Heidelberg, 2012.

[23] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[24] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in a social communication network. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 435–443, New York, NY, USA, 2008. ACM.

[25] Daniel Le Berre, Anne Parrain, M Baron, J Bourgeois, Y Irrilo, F Fontaine, F Laihem, O Roussel, and L Sais. Sat4j-a satisfiability library for java, 2006.

[26] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33(1):387–401, 2000.

[27] Yogesh S. Mahajan, Zhaohui Fu, and Sharad Malik. Zchaff2004: An efficient sat solver. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing*, SAT'04, pages 360–375, Berlin, Heidelberg, 2005. Springer-Verlag.

[28] João P Marques-Silva and Karem A Sakallah. Grasp: A search algorithm for propositional satisfiability. *Computers, IEEE Transactions on*, 48(5):506–521, 1999.

[29] Matthew W. Moskewicz, Conor F. Madigan, and Sharad Malik. Method and system for efficient implementation of boolean satisfiability, August 26 2008. US Patent 7,418,369.

[30] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 530–535, New York, NY, USA, 2001. ACM.

[31] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

[32] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning sat solvers with restarts. In *Principles and Practice of Constraint Programming-CP 2009*, pages 654–668. Springer, 2009.

[33] Nicola Santoro, Walter Quattrociocchi, Paola Flocchini, Arnaud Casteigts, and Frédéric Amblard. Time-varying graphs and social network analysis: Temporal indicators and metrics. *CoRR*, abs/1102.0629, 2011.

[34] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[35] Philip D Straffin. Linear algebra in geography: Eigenvectors of networks. *Mathematics Magazine*, 53(5):269–276, 1980.

[36] Philip S. Yu, Xin Li, and Bing Liu. Adding the temporal dimension to search - a case study in publication search. In Andrzej Skowron, Rakesh Agrawal, Michael Luck, Takahira Yamaguchi, Pierre Morizet-Mahoudeaux, Jiming Liu, and Ning Zhong, editors, *Web Intelligence*, pages 543–549. IEEE Computer Society, 2005.

8 Appendix

For the purposes of the discussion below, consider a dynamically evolving TVIG G of variables/clauses of a Boolean formula F input to a SAT solver. The graph G evolves as the solver learns new clauses and adds it to the learnt clause database. Further assume that G is modified only through addition of nodes/edges (correspondingly, variables/clauses in F), without any deletions or morphing of nodes/edges (correspondingly, no clause deletion or in-processing). Further assume that the cVSIDS ranking is initialized to the ranking produced by the TDC measure over the TVIG of the input formula F. Under these assumptions, the cVSIDS ranking of F and learnt clauses, at any point during the solving process, will exactly match the TDC rankings of the nodes of the dynamically evolving TVIG G of F and learnt clauses. We prove this formally below.

For the theorem below, we use the following definitions, notations and conventions. The goal is to abstract away those inner-workings of CDCL SAT solvers that are irrelevant here.

8.1 Definitions

Ranking: of variables (resp. nodes) in a Boolean formula F (resp. TVIG G): Consider a set of pairs, where the first element is a variable in F (resp. corresponding node in G), and the second element is a real number. A ranking R(F) over F (resp. R(G) over G) is a decreasing order over such pairs based on the decreasing order over the real numbers in these pairs. We say two rankings R_1 and R_2 match (or are equal), written as $R_1 = R_2$, if every i^{th} element of R_1 and R_2 are pairwise equal.

Linearly-ordered Family of TVIG graphs: We say $G = g_0, g_1, g_2, ...$ represent a linearly-ordered family of TVIG for a corresponding family $F = f_0, f_1, f_2, ...$ of Boolean formulas, if the following conditions hold:

- g_i is the TVIG of f_i
- f_i (resp. g_i) is obtained from f_{i-1} (resp. g_{i-1}) only by adding a single clause c (resp. edges among the nodes corresponding to c in g_{i-1}), without any other modifications/deletions to existing clauses in f_{i-1}.
- For all i, f_i share the same variables.

The cVSIDS Function: We need a precise treatment of cVSIDS heuristic in order to prove the theorem. We say that the cVSIDS function takes as input a formula f_i and the ranking of the variables over the formula f_{i-1}. Every variable in f_i get assigned the ranking of the corresponding variable in f_{i-1}. Additionally, cVSIDS bumps by 1 the ranks of the variables of the clause added to f_{i-1} to obtain f_i, and multiplicatively decays the ranks of all variables. It then orders the ranks and outputs it.

8.2 The Proof of cVSIDS Centrality Theorem

Theorem 1. *Let $R_{cVSIDS}(f_i)$ denote the ranking produced by the cVSIDS function over the formula f_i. Similarly, let $R_{TDC}(g_i)$ denote the ranking produced*

by the TDC measure over the graph g_i. If $R_{cVSIDS}(f_i) = R_{TDC}(g_i)$ then for all $j > i$ $R_{cVSIDS}(f_j) = R_{TDC}(g_j)$.

Proof. (by Induction)

We prove the theorem by by induction on i.

Base case: We note that the variant of cVSIDS we consider initializes the ranking (i.e., $i = 0$) of variables in f_0 with the TDC ranking over g_0.

Inductive Hypothesis: Assume that $R_{cVSIDS}(f_i) = R_{TDC}(g_i)$.

Inductive Step:

Let c be a clause added to f_i to obtain f_{i+1}. Correspondingly, we add appropriate edges to the TVIG g_i to obtain g_{i+1}. Let v be a variable in c, and the corresponding node in g_i is also denoted as v. In the TVIG g_{i+1}, the node v gains $|c| - 1$ more edges over the corresponding node v in g_i, each of weight $\frac{1}{|c|-1}$. This adds 1 to the TDC of v. Therefore the TDC of v is increased by 1. Likewise, cVSIDS bumps the activity of v (and other variables in the added clause) by 1. Hence cVSIDS and TDC increases activity/centrality by the same amount on each new learnt clause.

cVSIDS then decays the activities by multiplying by a constant decay factor α, where $0 < \alpha < 1$. Likewise, the TVIG decays the weights of the edges by the same decay factor. Therefore the equality between the activity score of variables and their TDC is maintained. □

This theorem analytically explains the correlation between cVSIDS and TDC. Although SAT solvers typically set the initial activites of variables to zero, we seed them to be the same as the TDC in this Section. This enables us to prove the theorem.

We do not consider clause deletion and in-processing while proving the above theorem. The reason is that these two operations modify the TVIG of the original+learnt clauses, thus introducing discrepancies between cVSIDS and TDC that are difficult to account for analytically. Having said that, in the paper we experimentally showed that cVSIDS and TDC are highly correlated even in the presence of clause deletion and in-processing.

In the following we empirically verify the theorem, and thus show that, under our assumptions, the theorem is relevant in practice.

8.3 Experiment: cVSIDS Centrality Theorem

	cVSIDS vs TDC
Mean top-10	0.977
Mean Pearson	0.998

Table 6. Results of cVSIDS Centrality Theorem experiment.

In this experiment, clause deletion and in-processing are switched off and the initial cVSIDS activity scores are initialized with the initial TDC of the original clauses. The idea is to control the experiment to replicate the conditions of the

cVSIDS Centrality Theorem as closely as possible to empircally reinforce the theorem. The data shows an almost perfect Pearson correlation of 0.998. Also 0.977 of the top-ranked cVSIDS variables are also highly ranked in TDC. The reason the metrics are not equal to 1.000 is likely due to the loss of precision from working with very large floating point numbers. The Sat4j implementation of VSIDS is based off the scheme introduced by MiniSAT, coined by Biere as EVSIDS [7]. Instead of decrementing the activity scores of each variable on each decay, Sat4j will multiplicatively increase the bump amount for all future bumps. Once the activity scores exceed 10^{100}, all the scores will be scaled down by a factor of 10^{100}. Arithmetic on large 64-bit floating point numbers will lead to a loss of precision.