Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers

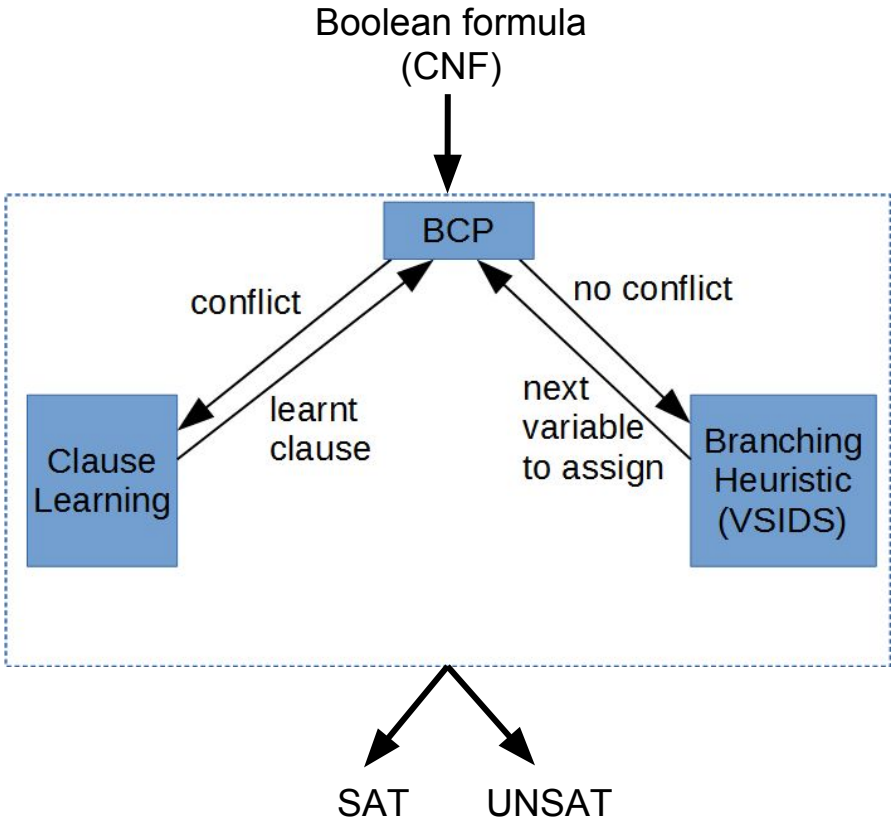
Jia Hui Liang, Vijay Ganesh, Ed Zulkoski, Atulan Zaman, Krzysztof Czarnecki

November 18, HVC 2015

Informal Problem Statement

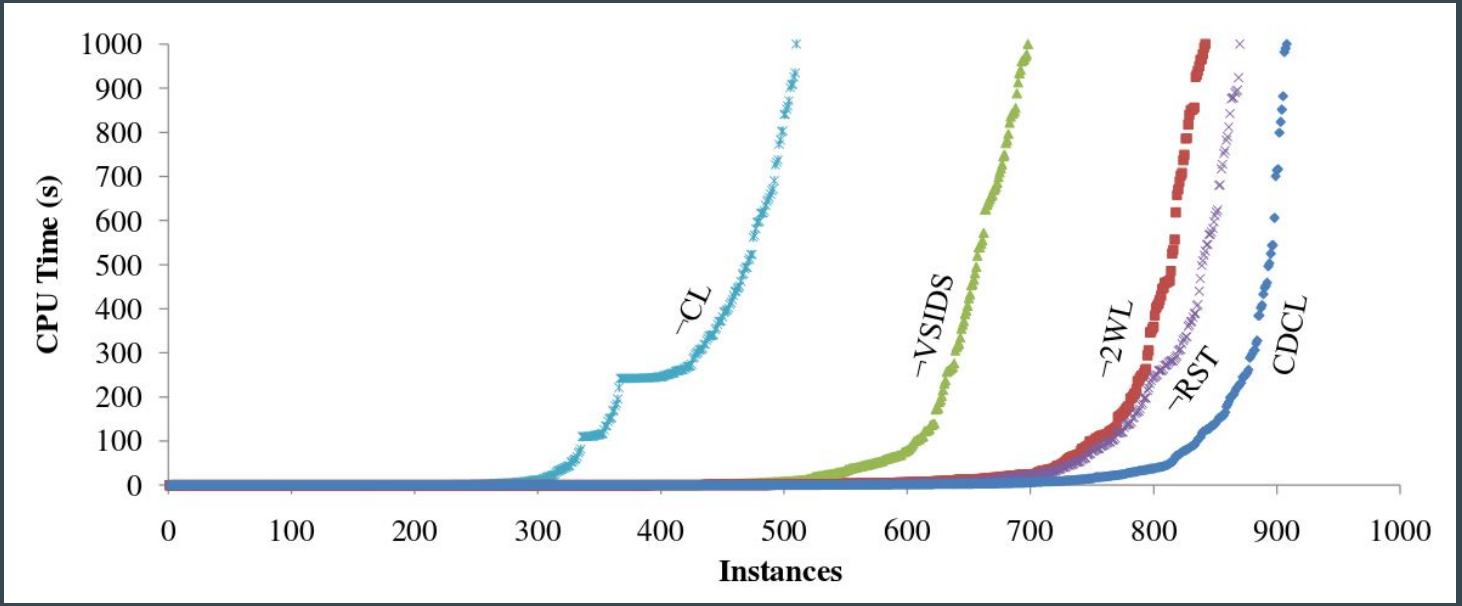
1. Discover the empirical invariants of VSIDS branching heuristic
2. Leverage this insight to construct better heuristics
3. But first, a little bit of background and motivation

Conflict-driven Clause-learning (CDCL)



Motivation for Studying Branching Heuristics

Katebi et al. (2011). Empirical Study of the Anatomy of Modern SAT Solvers.



Problem Statement

1. Why does VSIDS decay?
2. Which variables does VSIDS bump?
3. Does VSIDS focus on (local) regions of the search space?

State-of-the-Art Branching Heuristic: VSIDS

History

Proposed by the authors of the Chaff solver in 2001.

- Give more weight to recent conflicts.
- Little computational overhead.
- Implemented by most competitive CDCL solvers.

Bumping

Every variable has a floating-point number called “activity” score.

- Activity scores are initialized to zero.
- Add one (“bump”) to all the activities of variables occurring in conflict analysis.

Decaying

Recent conflict analysis is more pertinent to the current state of the solving.

- Multiply the activities of every variable by $0 < \delta < 1$ after every conflict (“decay”).

Conflict Timeline (δ = 0.5)

t=1	t=2	t=3	t=4	t=5	t=6
X	X		X		
0.5	0.75	0.37	0.68	0.34	0.17
		Y			Y
0	0	0.5	0.25	0.12	0.56

Pick X or Y as the next branching variable?

- The variable X was in more conflicts than Y.
  Perhaps we should focus on X?
- However, variable Y was in more **recent** conflicts.
  VSIDS picks Y over X.

Questions

1. Why does VSIDS decay?

2. Which variables does VSIDS bump?

3. Does VSIDS focus on (local) regions of the search space?

Findings

1. Exponential moving average (reinforcement learning)
2. Bridge variables in VIG and temporal graph centrality
3. Community structure focus

Result 1: VSIDS decay is an exponential moving average (reinforcement learning)

Noisy Time-Series and Smoothing

Exponential Moving Average

Brown's simple exponential smoothing

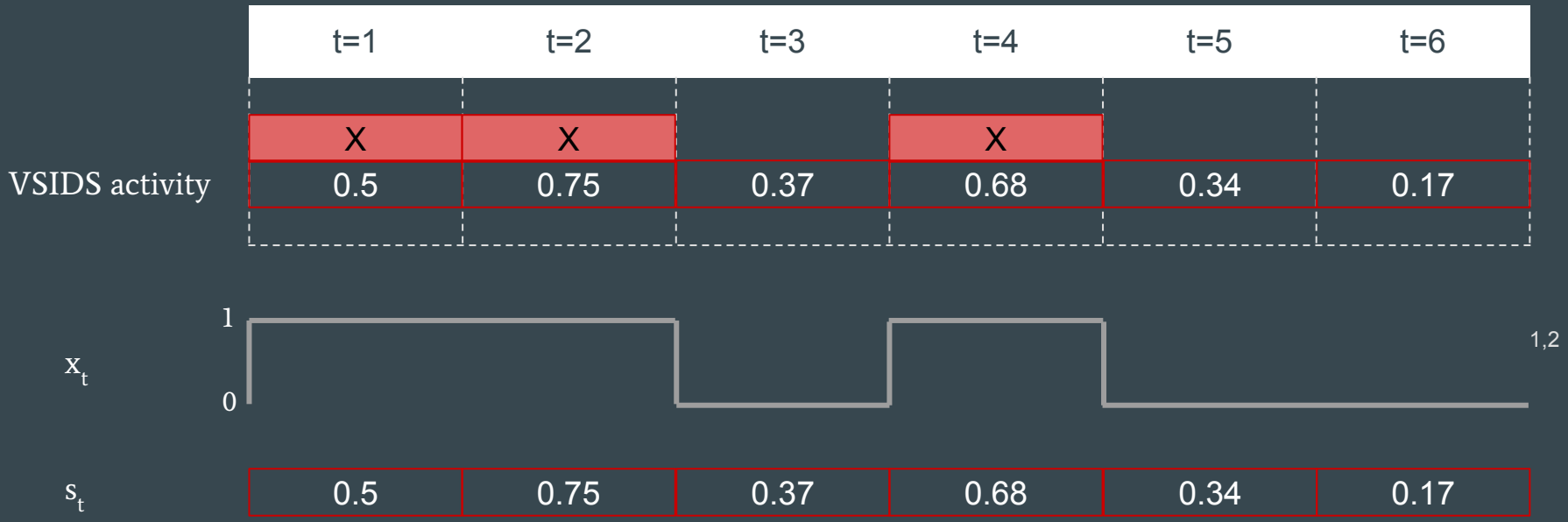
x_t is the time-series data (i.e., price of a stock).

Brown's simple exponential smoothing is defined as:

$$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{t-1}$$

α is the smoothing factor

Conflict Timeline (α = 0.5)



$s_0 = 0$

$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{t-1}$

[1] Biere, Armin. (2008). Adaptive Restart Strategies for Conflict Driven SAT Solvers.
[2] Huang et al. (2012). SAS+ Planning as Satisfiability.

Why use exponential moving averages?

1. **Financial analysis**: reduce random noise
2. **Signal processing**: low-pass filtering to remove high frequency noise
3. **Reinforcement learning**: learning from rewards in non-stationary environments

“Exponential Recency Weighted Average Branching Heuristic for SAT Solvers”

AAAI 2016

We took a well-known reinforcement learning algorithm for the non-stationary multi-armed bandit problem and used it as a branching heuristic.

Improvement on MiniSat with CHB

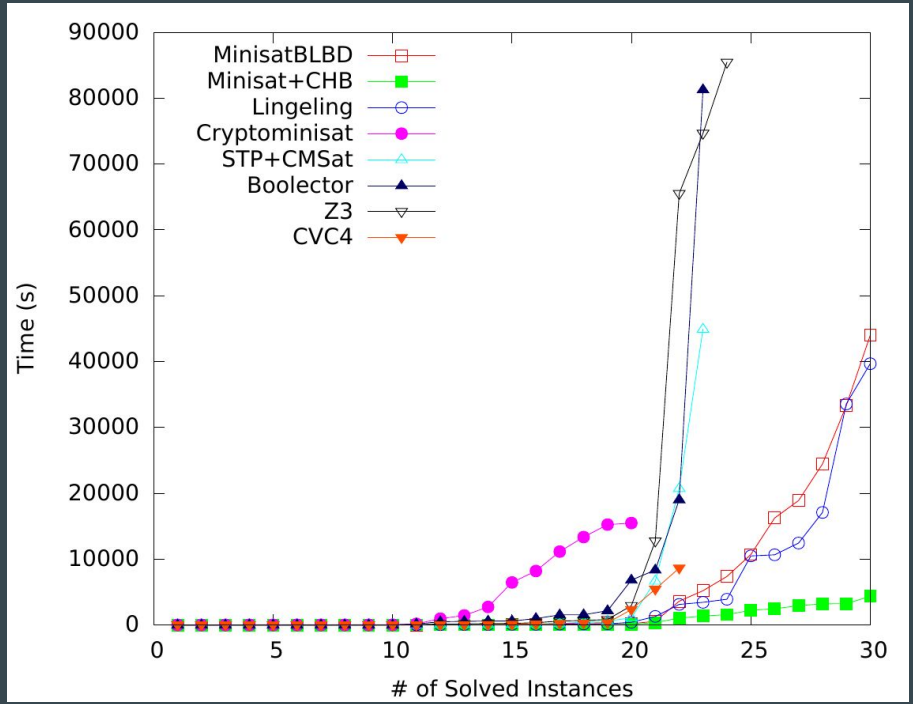
The number “+x (+y%)” means CHB is solving x more instances than VSIDS, an improvement of y percent.

	Improvement with CHB
2013 + 2014 Application	+33 (+10.3%)
2013 + 2014 Crafted	+64 (+22.5%)

For the satisfiable application track for 2013 and 2014, MiniSat with CHB solves more instances than the respective winners.

SHA-1 Preimage Attack

Preimage attack on step-reduced SHA-1 instances.



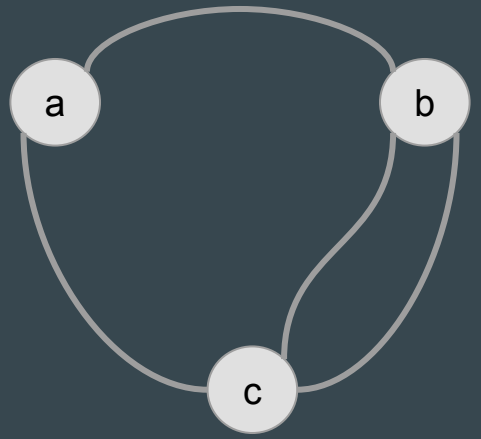
Result 2: VSIDS picks, bumps, and learns over bridge and temporally central variables

Variable Incidence Graph (VIG) of a CNF

CNF:

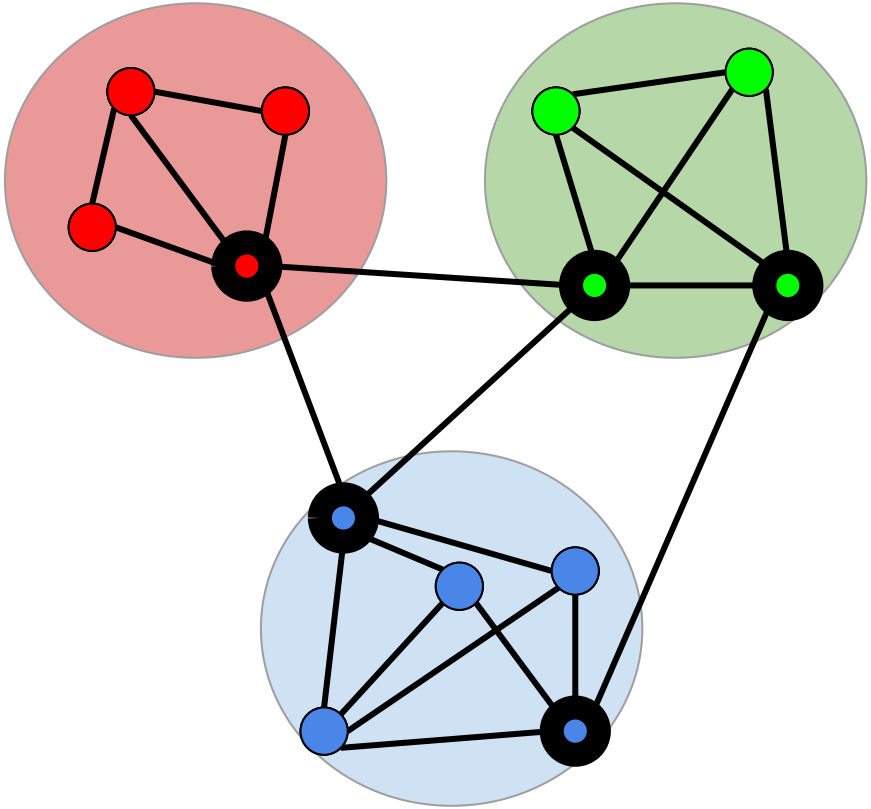
a or b or not c

not b or c



Community Structure of a Graph

- Partition the variables into “communities.”
- **Modularity(q)**: many internal edges, few external edges.
- **Bridge variable**: adjacent to an external another variable.

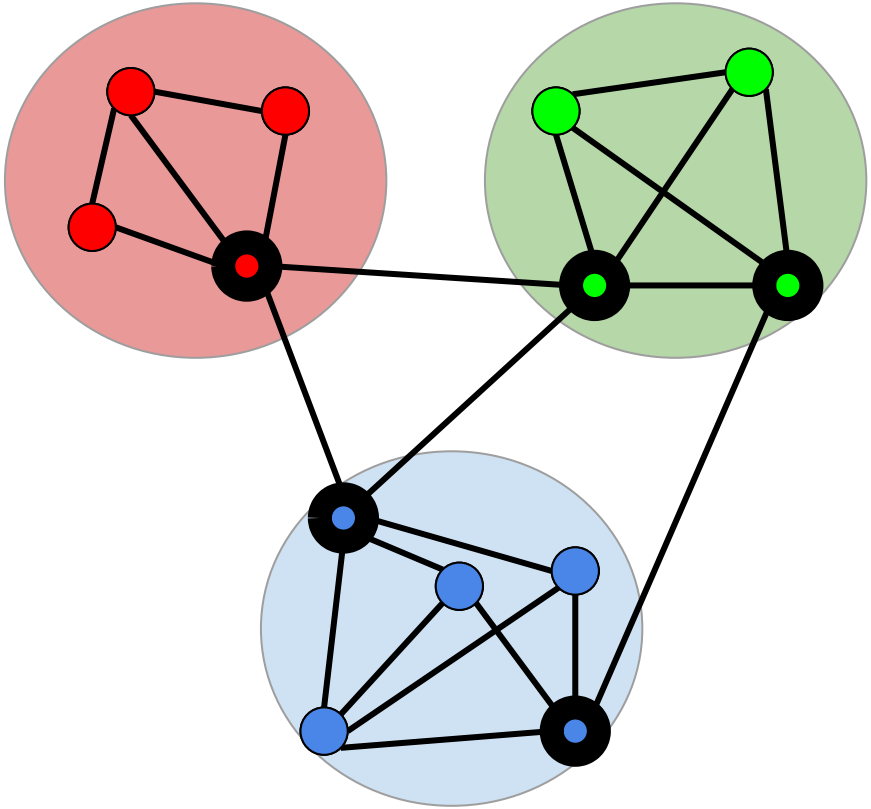


“[The] community structure of real-world industrial instances is a better predictor of the running time of CDCL solvers than other commonly considered factors such as variables and clauses.” [1]

[1] Newsham et al. (2014). Impact of Community Structure on SAT Solver Performance.

Bridge Variables

- Bridge variables connect communities together.
- Related to betweenness centrality, “a measure of the influence a node has over the spread of information through the network.” [2]



[2] Newman, M. E. (2005). A measure of betweenness centrality based on random walks.

Affinity for Bridge Variables

Category from SAT Competition 2013	% of variables in CNF that are bridge	% of VSIDS picks that are bridge	% of VSIDS bumps that are bridge	% of learnt clause variables that are bridge
Application	61.0%	79.9%	71.6%	78.4%

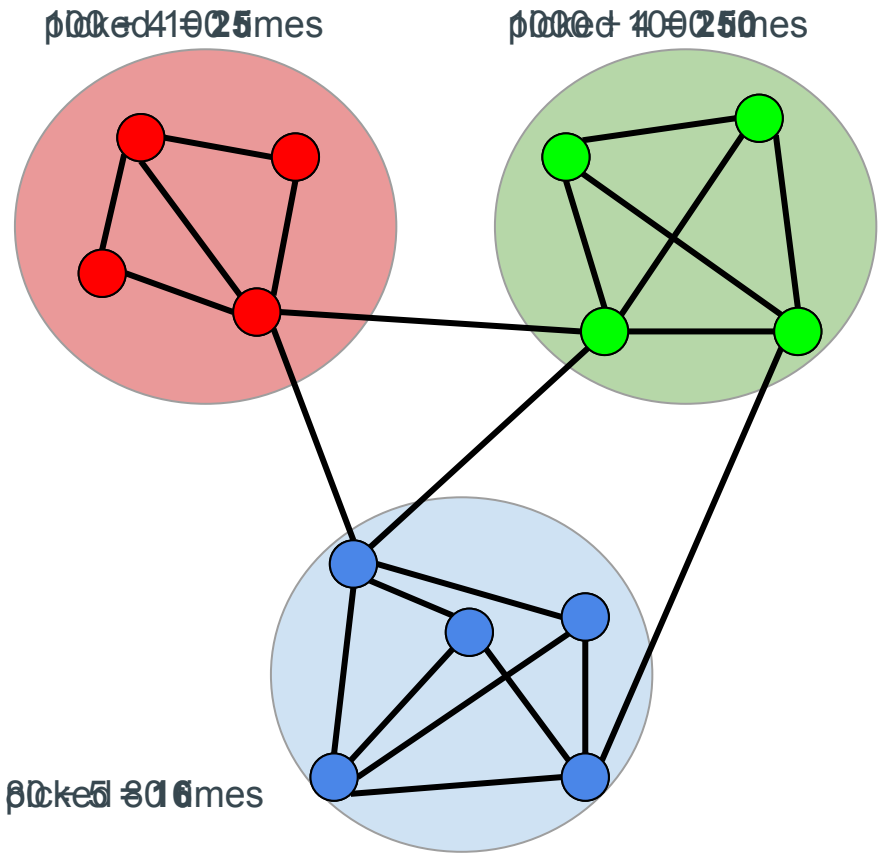
Result 3: VSIDS performs focused search over the community structure of the input formula

“Conceptually the idea is to focus on those variables, that have the strongest influence on the current region of the search space.” [3]

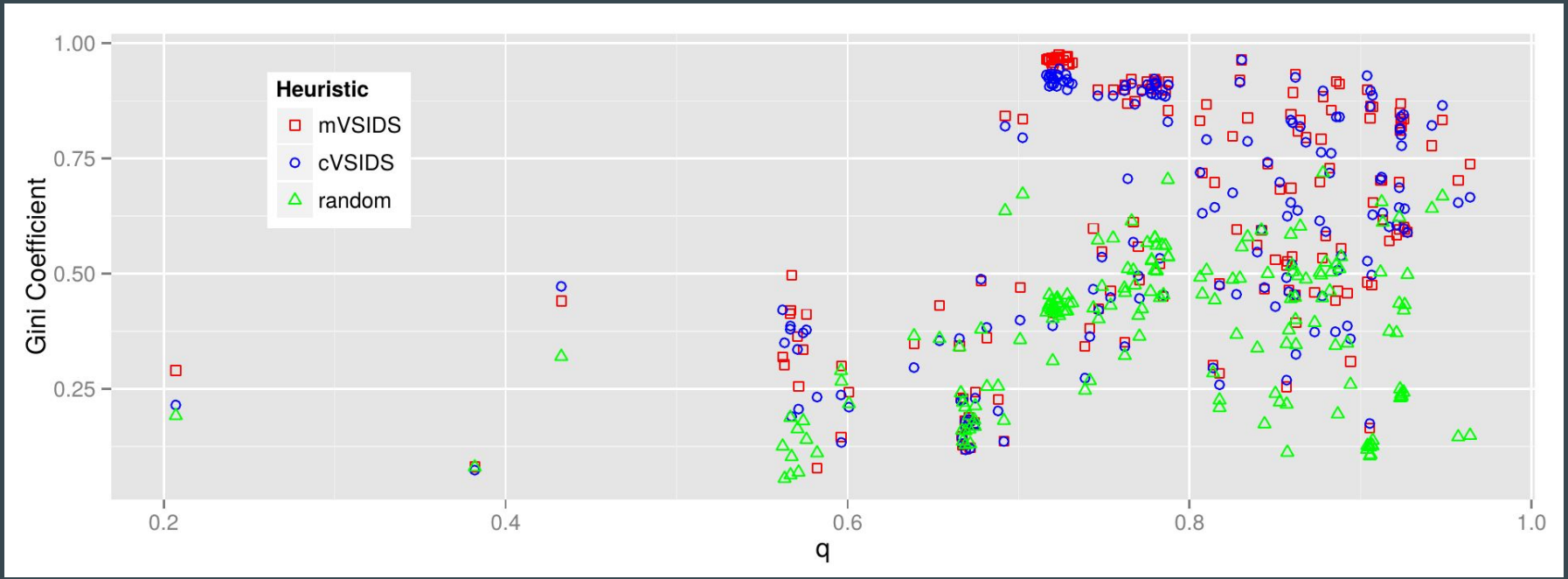
[3] Shi et al. (2005). PASSAT: Efficient SAT-based Test Pattern Generation for Industrial Circuits.

Spatial

- Certain regions (i.e., communities) are favoured.
- **Community score**: number of VSIDS picks from the community normalized by the order of the community.
- Compute GINI coefficient of the community scores.
- GINI closer to 1: inequality
- GINI closer to 0: equality



Spatial (Industrial)

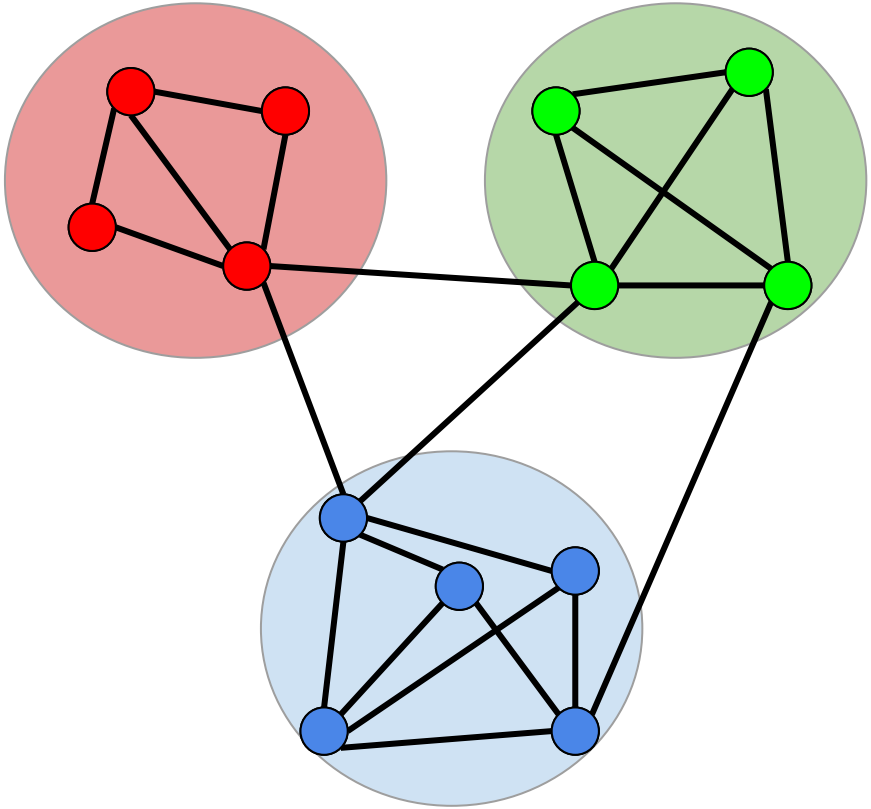


Average GINI: mVSIDS 0.592, cVSIDS = 0.560, random = 0.216

Temporal

Similar to temporal locality in hardware cache and cache hit rate.

More details in the paper.



Understanding VSIDS: Conclusions

Questions we posed

1. Why does VSIDS decay?
2. Which variables does VSIDS bump?
3. Does VSIDS focus on (local) regions of search space?

Our Findings

1. Exponential moving average (reinforcement learning)
2. Bridge variables in the VIG and temporal graph centrality
3. Community structure focus

Interpretations

1. Reduce noise and reinforcement learning (AAAI 2016 and SHA-1)
2. Highly central variables and bridge variables may coincide with backdoors
3. Community locality implies search locality and cache locality