# WAPS – Weighted and Projected Sampling

Rahul Gupta*, Shubham Sharma*, Subhajit Roy*, Kuldeep S. Meel†
*Indian Institute of Technology Kanpur, †National University of Singapore

## 1. Motivation

- Software and hardware verification is performed using test vectors.
- Test vectors are obtained from a set of constraints according to user defined weight distribution and triggers.
- Applications in other domains: probabilistic reasoning, bug synthesis, statistical physics, machine learning, etc.

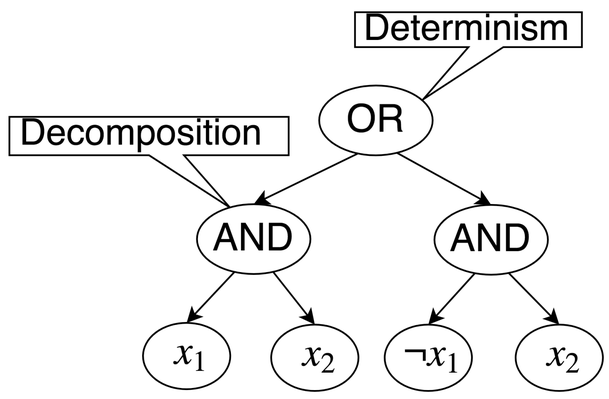
## 2. Problem Formulation

Given a CNF formula $F$, set of projecting variables $P$ and weight function $W(\cdot)$ over literals, build a weighted and projected sampler $\mathsf{WAPS}(F, P, W)$ such that,

$$\forall y \in R_{F\downarrow P}, \mathsf{Pr}\left[y = \mathsf{WAPS}(F, P, W)\right] = \frac{W(y)}{W(R_{F\downarrow P})}$$

where weight of an assignment is the product of weight of it's literals.

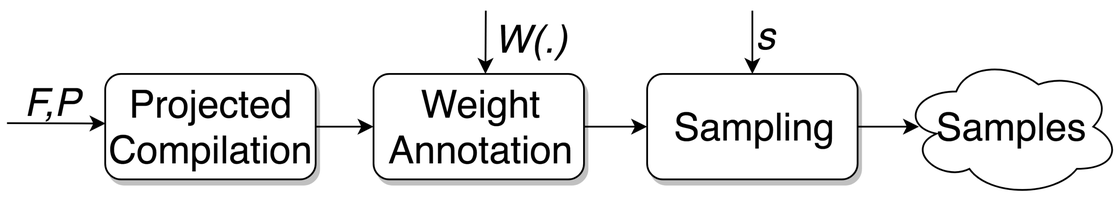
## 2. d-DNNF representation




## 3. Previous Work

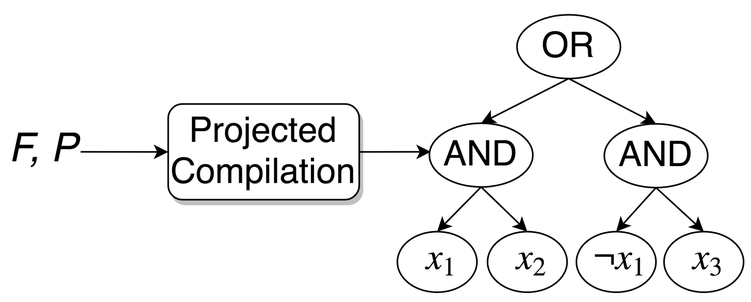
- WeightGen [Chakraborty et al. 2014]: Hashing based technique restricted to low tilt weights.
- KUS [Sharma et al. 2018]: Uniform sampling using precompiled d-DNNF.

## 4. WAPS

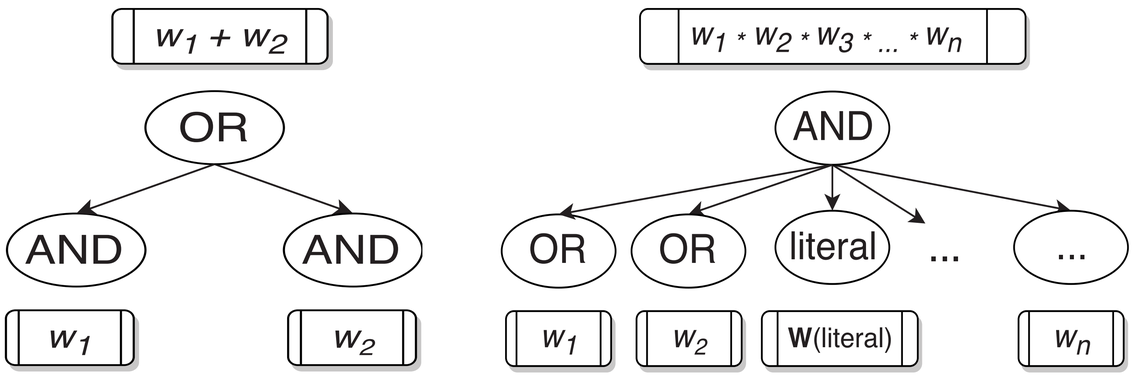



## 5. Projected Compilation

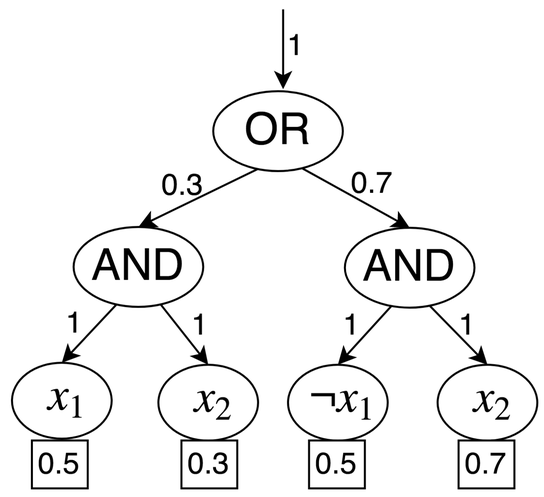
$F = (x_6 \vee x_5 \vee \neg x_1 \vee x_3) \wedge (x_3 \vee x_6 \vee \neg x_5 \vee \neg x_1) \wedge (\neg x_2 \vee x_4 \vee \neg x_1) \wedge (x_1 \vee x_2), (x_3 \vee \neg x_6 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_5 \vee x_6)$
$P = \{x_1, x_2, x_3\}$




## 6. Bottom-Up Weight Annotation

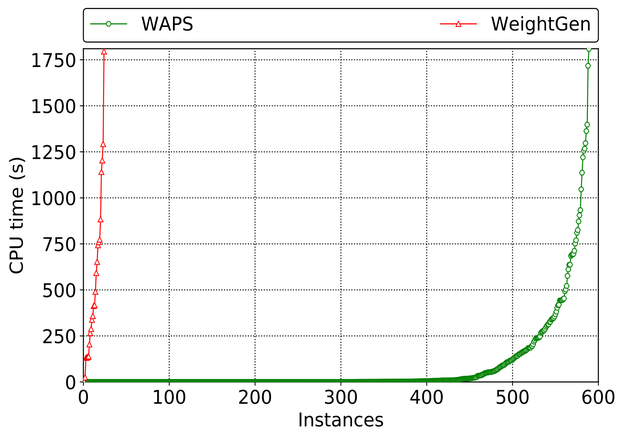


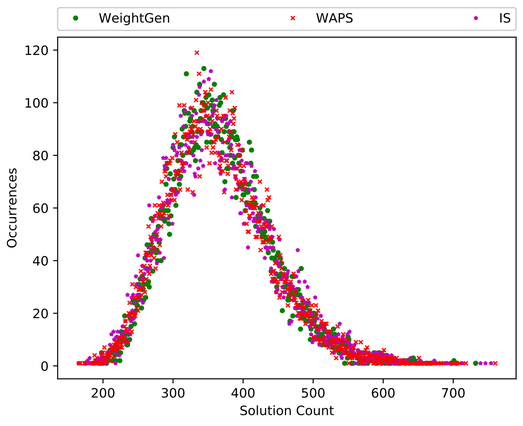

## 7. Sampling

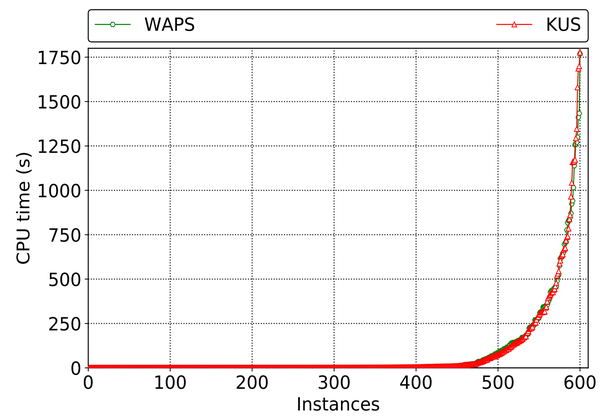


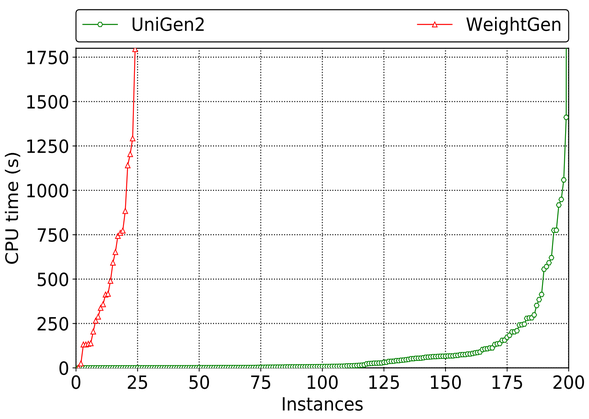

## 8. Evaluation

WAPS performance comparison (Timeout = 1800s)




| Benchmark | Vars | Clauses | $\lvert P\rvert$ | WeightGen | WAPS | | | WeightGen / WAPS |
|---|---|---|---|---|---|---|---|---|
| | | | | | Compile | A+S | Total | |
| s526_15_7 | 452 | 1303 | 22 | 652.48 | 91.66 | 31.15 | 122.81 | 5.31 |
| s526a_3_2 | 366 | 944 | 24 | 490.34 | 15.37 | 1.96 | 17.33 | 28.29 |
| LoginService | 11511 | 41411 | 36 | 1203.93 | 15.02 | 0.75 | 15.77 | 76.34 |
| blockmap_5_2 | 1738 | 3452 | 1738 | 1140.87 | 0.04 | 5.30 | 5.34 | 213.65 |
| s526_3_2 | 365 | 943 | 24 | 417.24 | 0.06 | 0.67 | 0.73 | 571.56 |
| or-50-5-9-UC-40 | 100 | 250 | 100 | 743.1 | 0.01 | 0.41 | 0.42 | 1769.29 |
| or-100-5-4-UC | 200 | 500 | 200 | 1795.52 | 0.01 | 0.74 | 0.74 | 2426.38 |
| or-50-5-10-UC | 100 | 250 | 100 | 1292.67 | 0.01 | 0.36 | 0.36 | 3590.75 |
| blasted_case35 | 400 | 1414 | 46 | TO | 0.57 | 1.46 | 2.03 | - |
| or-100-20-4-UC | 200 | 500 | 200 | TO | 0.19 | 2.48 | 2.67 | - |

Table 1: Run time (in seconds) for 1000 samples

## 9. Conclusion

- WAPS outperformed WeightGen by up to 3 orders of magnitude in runtime while achieving a geometric speedup of 296×.
- The distribution generated by WAPS is statistically indistinguishable from that generated by an ideal weighted and projected sampler.
- The performance of WAPS is oblivious to the weight distribution.

- Come to our talk, 12:00 PM - 12:30 PM on $8^{th}$ April as part of TACAS 2019.