Using Community Structure to Detect Relevant Learnt Clauses

Carlos Ansótegui(1), **Jesús Giráldez-Cru**(2), Jordi Levy(2), and Laurent Simon(3)

(1) DIEI, Universitat de Lleida, Spain

(2) **Artificial Intelligence Research Institute, Spanish National Research Council** (**IIIA-CSIC**), Barcelona, Spain

(3) LaBRI, Université de Bordeaux, France

SAT Conference 2015
September 25th, 2015

Overview

Introduction

- **Clause Learning** is the most important ingredient of CDCL SAT solvers to explain their **success** on solving industrial SAT instances.

  [KatebiSakallahMarques-Silva.SAT11]

- **Not** all learnt clauses have the same **relevance** or **usefulness** during the search.
  - The usefulness of a learnt clause may vary during the search.

- **Aggressive clause removal policies** are now an **essential** ingredient of CDCL solvers.

On the Relevance of Learnt Clauses

- **Original** instance:
  - solved after c conflicts.
- **Augmented** instance:
  - repeat the same execution,
  - stop the search after $p \cdot c$ conflicts $(0 < p < 1)$,
  - augment the original instance with the set of learnt clauses the solver is keeping at that instant.

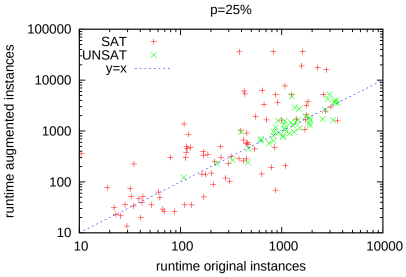
solving original instances

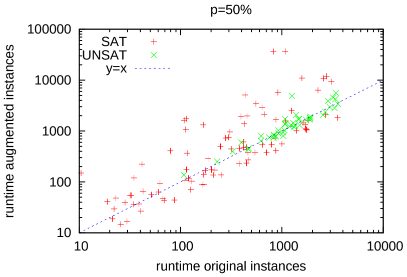
vs

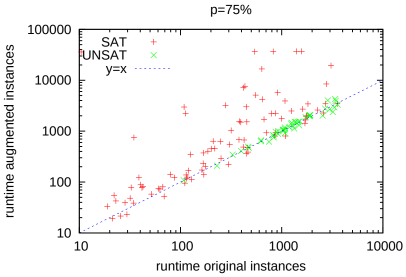
generating + solving augmented instances

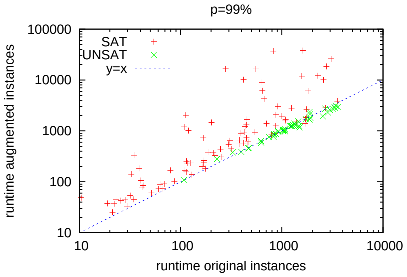
?

Experiment

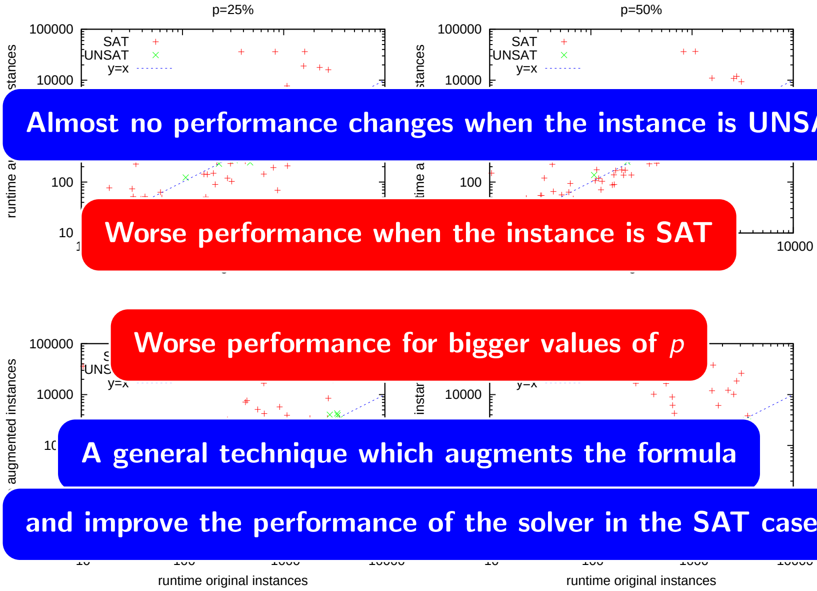

p=25%
runtime augmented instances
runtime original instances
SAT
UNSAT
y=x
10
100
1000
10000
100000


p=50%
runtime augmented instances
runtime original instances
SAT
UNSAT
y=x
10
100
1000
10000
100000


p=75%
runtime augmented instances
runtime original instances
SAT
UNSAT
y=x
10
100
1000
10000
100000


p=99%
runtime augmented instances
runtime original instances
SAT
UNSAT
y=x
10
100
1000
10000
100000

Experiment



Almost no performance changes when the instance is UNSAT

Worse performance when the instance is SAT

Worse performance for bigger values of p

A general technique which augments the formula

and improve the performance of the solver in the SAT case?

Related Work

- How to **measure** the usefulness of a learnt clause?

- **Literal Block Distance** (**LBD**):
  [AudemardSimon.IJCAI09]
  - Number of decision levels in the learnt clause.
  - Smaller is better.

- Database management implemented in **Glucose**:
  - Clauses with low LBD are useful.
  - **Aggressively remove** learnt clauses with high LBD.
  - Not only about maintaining good unit propagation rates.
  - To guide the solver to *easier* proofs.

Motivation

- LBD is **correlated** to the number of communities of learnt clauses.
  [NewshamGaneshFischmeisterAudemardSimon.SAT14]

- The previous observation is **one-way**:
  - From LBD to community structure.

- Is it possible to **exploit** this correlation in **the other way**: using the community structure to detect relevant learnt clauses?

Contributions

- The **community structure** can be **used to detect relevant learnt clauses**.
- Implemented in a **fast preprocessing step** (**modprep**) augmenting the original formula:
  - On **UNSAT**, it does not worse the performance of the solver, or it even improves its performance.
  - On **SAT**, it improves the performance of several solvers:
    - **MiniSAT**: a popular CDCL solver.
    - **Glucose**: good on **UNSAT** instances.
      2nd classified SAT Competition 2014, UNSAT category
    - **MiniSAT-blbd**: good solver for **SAT** instances.
      1st classified SAT Competition 2014, SAT category
    - **Lingeling**: good on both **SAT** and **UNSAT** instances.
      1st classified SAT Competition 2014, UNSAT category 1st classified SAT Competition 2014, SAT+UNSAT category

Overview

- Introduction
- **Community Structure**
- Proposed Solution
- Conclusions

The Community Structure of Graphs

- A graph has clear **community structure** if its nodes can be grouped into communities such that its edges mostly connect nodes of the same community.
- The **modularity** Q of a graph G and a partition C of its nodes measures the *fraction of internal edges* (w.r.t. to a random graph with same nodes and same degrees). [NewmanGirvan.PhysRev04]

- The **modularity** of a graph is the **maximal** modularity for any possible partition: $Q(G) = max\{Q(G,C)|C\}$.
- The (**optimal**) modularity ranges in the interval $[0,1]$.

The Community Structure of Graphs

- A graph has clear **community structure** if its nodes can be grouped into communities such that its edges mostly connect nodes of the same community.
- The **modularity** Q of a graph G and a partition C of its nodes measures the *fraction of internal edges* (w.r.t. to a random graph with same nodes and same degrees). [NewmanGirvan.PhysRev04]

- The **modularity** of a graph is the **maximal** modularity for any possible partition: $Q(G) = max\{Q(G, C) | C\}$.
- The (**optimal**) modularity ranges in the interval $[0, 1]$.

Computing the Modularity

- Computing the (optimal) modularity is **NP-hard**.
- Instead, most methods compute a **lower-bound of Q**.

Some methods:

- *GN community structure* alg.: [Newman.PhyRev04]
  - -fast and +accurate.
- **Louvain method**: [BlondelGuillaumeLambiotteLefebvre.JStMec08]
  - $\pm$fast and +accurate.
- *Online Community Detection* alg.: [ZhangPanWuLi.IJCAI13]
  - +fast and -accurate.

The Community Structure of Industrial SAT Instances (I)

- **SAT Instances** as Graphs.

- The **Variable Incidence Graph** (**VIG**):
  - Nodes are variables.
  - Edges between two variables in the same clause.
  - Weights to consider the length of the clause: it gives the same relevance to all clauses.

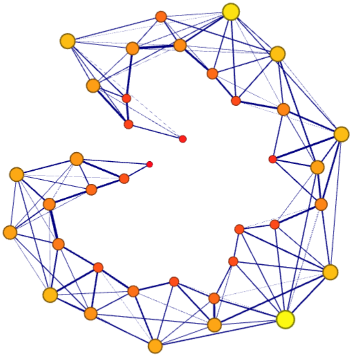
The Community Structure of Industrial SAT Instances (II)

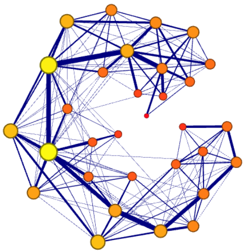
- **Industrial SAT instances** have a **clear community structure**.
  [AnsóteguiGiráldez-CruLevy.SAT12]

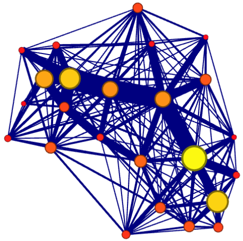
- Their **modularity** has values **greater than** 0.7 in most cases (**random** SAT instances have a modularity **smaller than** 0.3).

- The community structure is also **clear** if adding **learnt clauses**.
- However, the obtained **partition** may **change**.

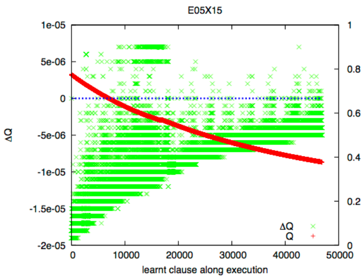
Clause Learning Destroys the (Orig.) Community Structure

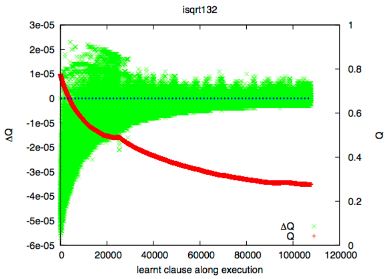

original structure


adding small learnt clauses


adding small and medium-size learnt clauses

Clause Learning Destroys the (Orig.) Community Structure





- If learnt clauses are considered, the community structure is still clear iff a new partition is recomputed.
- But using the original partition, most learnt clauses increase the modularity Q with $\Delta Q < 0$.
- Therefore, the **modularity** Q of the original partition **decreases**.

Overview

A Modularity-based SAT Instances Preprocessor

modprep Algorithm:

```
Input:    SAT Instance $\Gamma$
Output:   SAT Instance $\Gamma'$                // $\Gamma \subseteq \Gamma'$

1.   $\Gamma' := \Gamma$;
2.   $C := communityStructure(\Gamma)$; //Louvain method on VIG of $\Gamma$
3.   foreach pair $(c_i, c_j)$ of connected communities of $C$
4.       Solver $s$;
5.       $s.solve(c_i \cup c_j)$;
6.       if $(s == UNSAT)$
7.           return $\varnothing$;
8.       endif
9.       $\Gamma' := \Gamma' \cup s.learntClauses$;
10.  endforeach
11.  return $\Gamma'$;
```

The Cost of **modprep**

Computing the **community structure**:

- Average: 12.6s
- Median: 4.3s
- Max: 294.5s

Solving **all** the **subformulas**:

- Average: 78.0s
- Median: 21.8s
- Max: 975.8s

Number of **learnt clauses**:

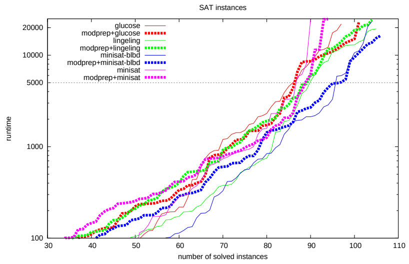
- Av.: 11243.9
- Median: 512.0
- Max: 794950
- Min: 1

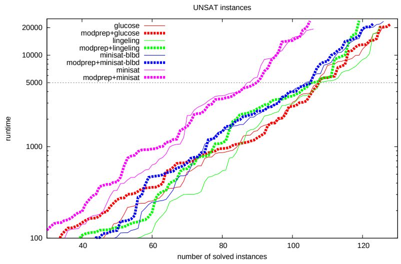
Discussion

- Why **pairs** of communities...?
- ...instead that tuples of **higher arities**?

- **LBD** correlated to the **number of communities**.
- The lower LBD, the **better**.
- **Balance** between the **cost** of **modprep** and the **relevance** of the obtained learnt clauses.

Experimental Evaluation (2011)

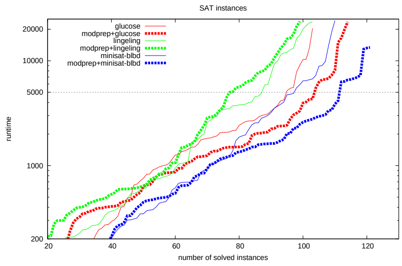


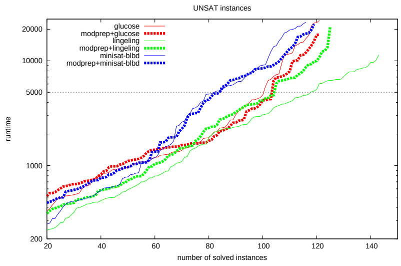


category	SAT				UNSAT			
timeout	5000s		25000s		5000s		25000s	
	orig	modpr	orig	modpr	orig	modpr	orig	modpr
MiniSAT	86	**88**	90	**94**	87	**89**	**106**	105
Glucose	85	**86**	97	**101**	107	***107**	126	**128**
MiniSAT-blbd	95	**96**	104	**106**	103	**105**	**126**	123
Lingeling	**89**	89	**105**	104	**107**	105	**125**	119

Performance of solvers on SATcomp2011, with/out **modprep**.

Experimental Evaluation (2014)

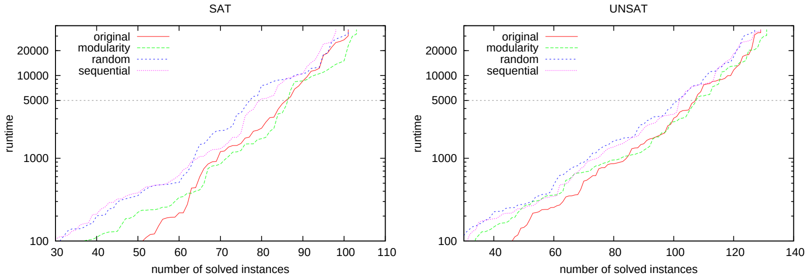




category	SAT				UNSAT			
timeout	5000s		25000s		5000s		25000s	
	orig	modpr	orig	modpr	orig	modpr	orig	modpr
Glucose	94	**103**	103	**114**	100	**103**	121	***121**
MiniSAT-blbd	97	**111**	110	**121**	81	**84**	116	**119**
Lingeling	**87**	77	**103**	99	**117**	104	**143**	125

Performance of solvers on SATcomp2014, with/out **modprep**.

Experimental Evaluation (Random)



category	SAT		UNSAT	
timeout	5000s	25000s	5000s	25000s
original	85	97	107	126
structure	**86**	**101**	***107**	**128**
random	77	96	101	122
sequential	79	96	101	122

Performance of Glucose on SATcomp2011, with random partitions.

Overview

Conclusions

- We use the **community structure** of industrial SAT instances to **identify** a set of **highly useful learnt clauses**.
- This is the set of clauses learnt from **solving** all subformulas consisting in **pairs of connected communities**.
- **Augmenting** the formula with this set of clauses **improves the performance** of the solver in many cases.
- This improvement is especially relevant in **satisfiable** instances.
- We obtain an overall improvement in **MiniSAT**, **Glucose** and **MiniSAT-blbd**.
- This is not the case in **Lingeling**.

Thank you
for your attention!

I am looking for postdocs opportunities from 2016
`jgiraldez@iiia.csic.es`