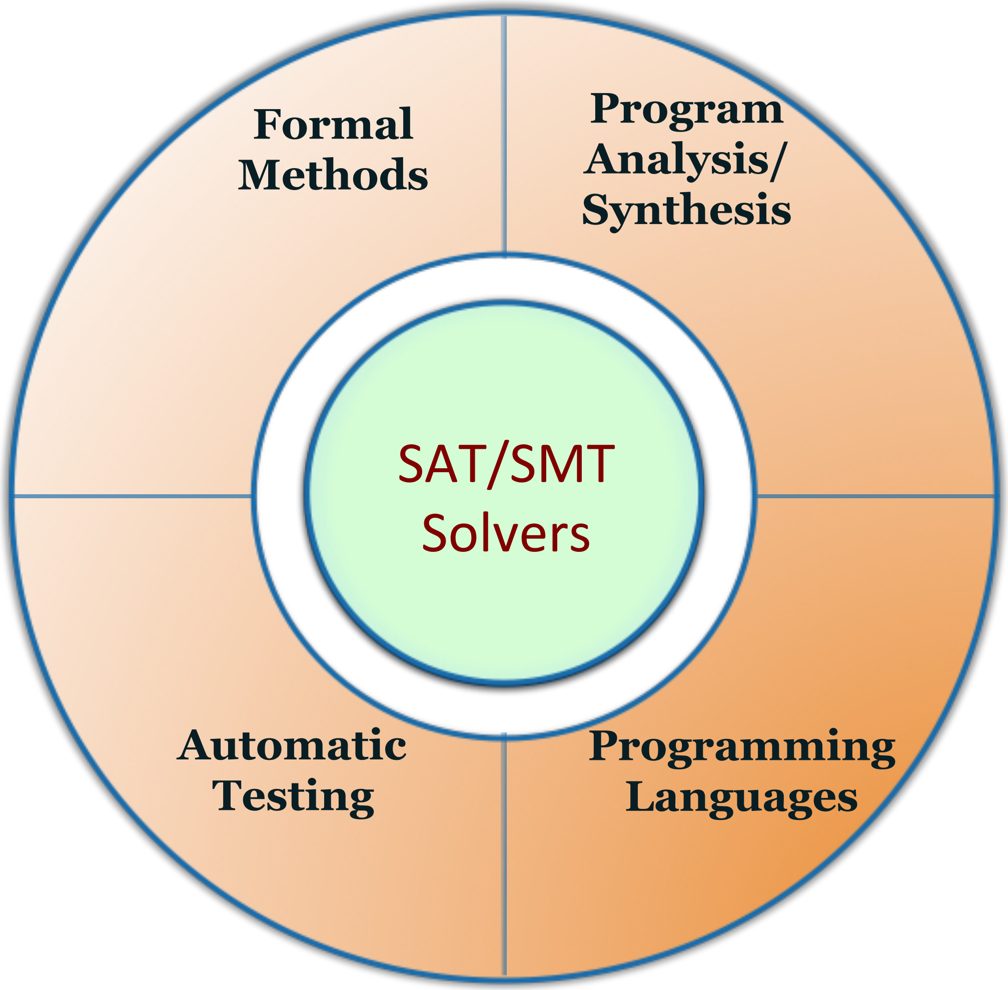
Impact of Community Structure on SAT Solver Performance*

by
Zack Newsham[1], Vijay Ganesh[1],
Sebastian Fischmeister[1], Gilles Audemard[2], and Laurent Simon[3]
[1]University of Waterloo, [2]University of Artois and [3]University of Bordeaux

Presented at SAT 2014, Vienna, Austria
(*Won the best student paper award)

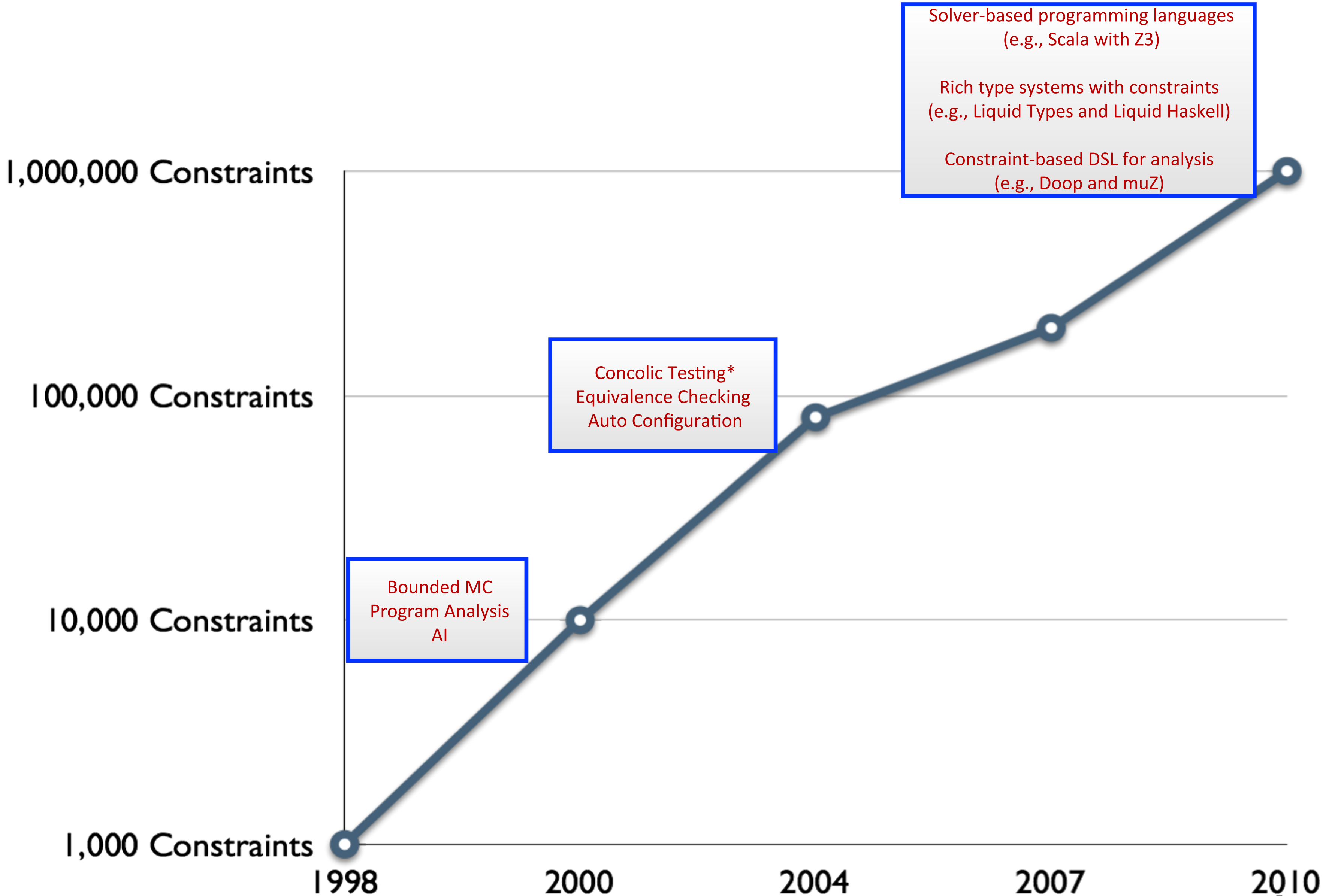
Software Engineering & SAT/SMT Solvers

An Indispensable Tactic for Any Strategy



SAT/SMT Solver Research Story

A 1000x Improvement in the Last Few Years



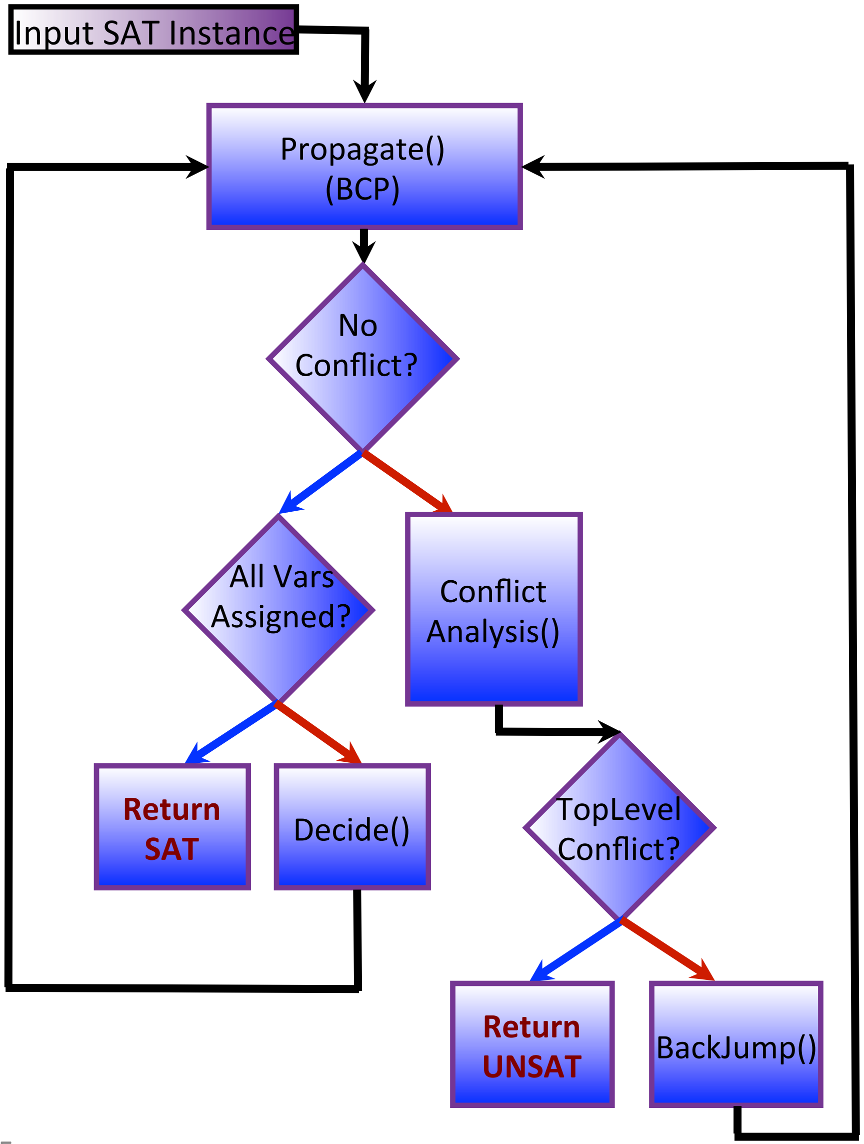
What is a SAT/SMT Solver?

Automation of Logic

- Rich logics (Modular arithmetic, Arrays, Strings,...)
- Boolean satisfiability problem is NP-complete, Quantified Boolean satisfiability problem is PSPACE-complete,...
- Practical, scalable, usable, automatic
- Enable novel software reliability approaches

Modern CDCL SAT Solver Architecture

Key Steps and Data-structures



Key steps

- Decide()
- Propagate() (Boolean constant propagation)
- Conflict analysis and learning() (CDCL)
- Backjump()
- Forget()
- Restart()

CDCL: Conflict-Driven Clause-Learning

- Conflict analysis is a key step
- Results in learning a learnt clause
- Prunes the search space

Key data-structures (Solver state)

- Stack or trail of partial assignments (AT)
- Input clause database
- Conflict clause database
- Conflict graph
- Decision level (DL) of a variable

Problem Statement

Why are SAT Solvers efficient for Industrial Instances

- Conflict-driven clause learning (CDCL) Boolean SAT solvers are remarkably efficient for large industrial instances

- This is true for industrial instances from a diverse set of applications

- These instances may have tens of millions of variables and clauses

- This phenomenon is surprising since Boolean satisfiability is an NP-complete problem believed to be intractable in general

- Why is this so?

Scientific Motivation to Understand Why SAT Works

The Laws of SAT Solving

- A scientific approach, as opposed to trial-and-error
- Lead to better, and more importantly predictable solvers
- Predictive model that cheaply computes solver running time by analyzing SAT input
- Complexity-theoretic understanding, a la smoothed analysis
- As yet unforeseen applications may benefit from a deeper understanding of SAT solving (more on this later)

The Laws of SAT Solving

Sub Problems

- We break the problem statement down to smaller sub-problems

1. On which class of instances do SAT solvers perform well? I.e., a precise mathematical characterization of instances on which solvers work well
2. An abstract algorithmic description of SAT solvers
3. A complexity-theoretic analysis that provides meaningful asymptotic bounds

In this talk, I focus on Question 1, and briefly touch upon some potential answers for Question 2.

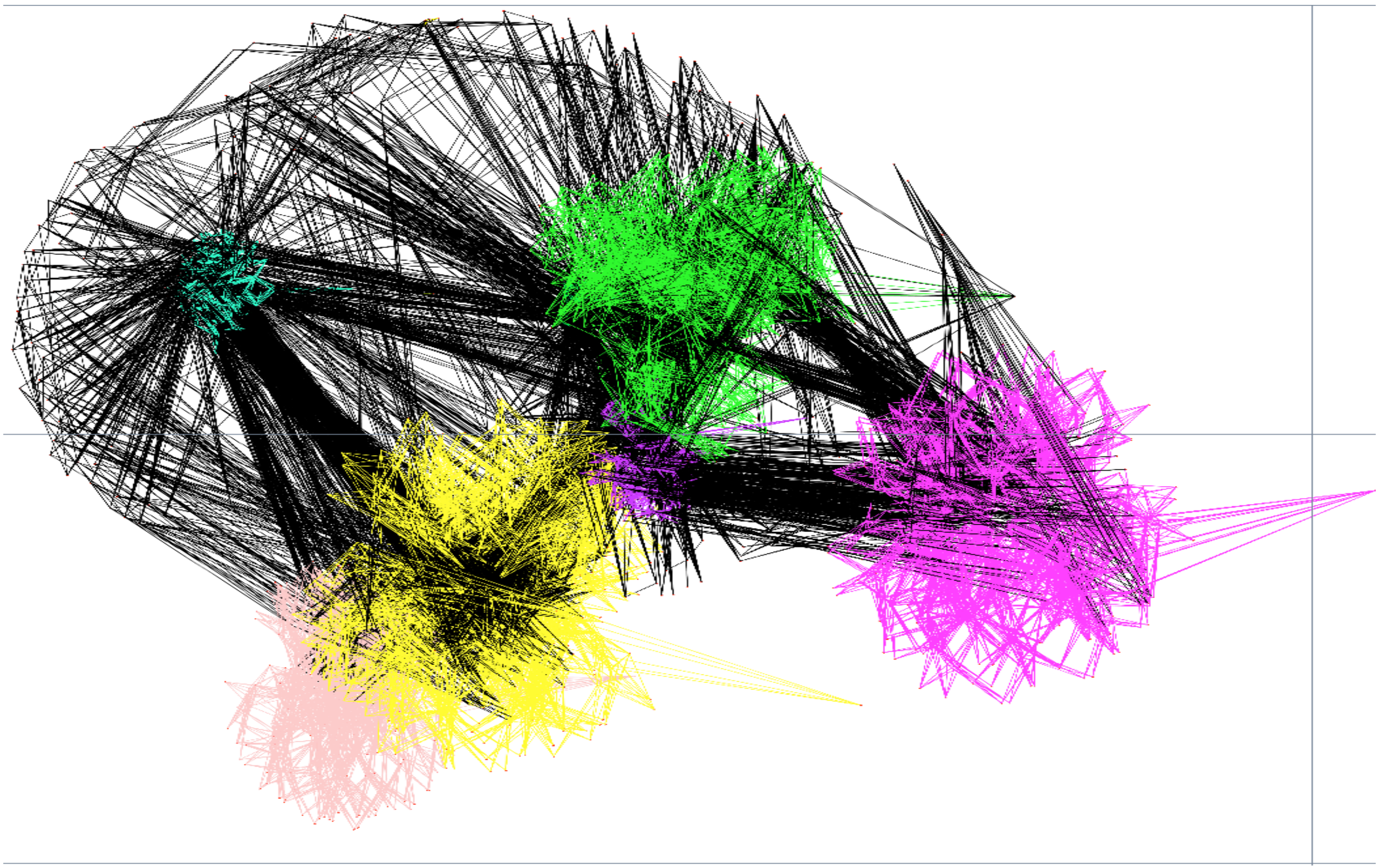
Community Structure and SAT Solver Performance
Our Results and Take-home Message

- A (partial) answer to question 1
  - A graph-theoretic characterization of SAT instances, as opposed to measuring the size of instances only in terms of number of variables and clauses
  - Industrial SAT instances have “good” community structure (also confirmed by previous work by Jordi Levy et al.)
  - Community structure of the graph of SAT instances strongly affect solver performance

- Result #1: Hard random instances have low Q ($0.05 \leq Q \leq 0.13$)
- Result #2: Number of communities and Q of SAT instances are more predictive of CDCL solver performance than other measures
- Result #3: Strong correlation between community structure and LBD (Literal Block Distance) in Glucose solver

SAT Formulas as Graphs

Boolean Variables are Nodes, Clauses are Edges



SOURCE: mrpp example from SAT 2013 competition viewed using our SATGraf tool

Community Structure in Graphs
Definition and Applications

- Community structure [GN03,CNM04,OL13] of a graph is measure of “how separable or well-clustered the graph is”

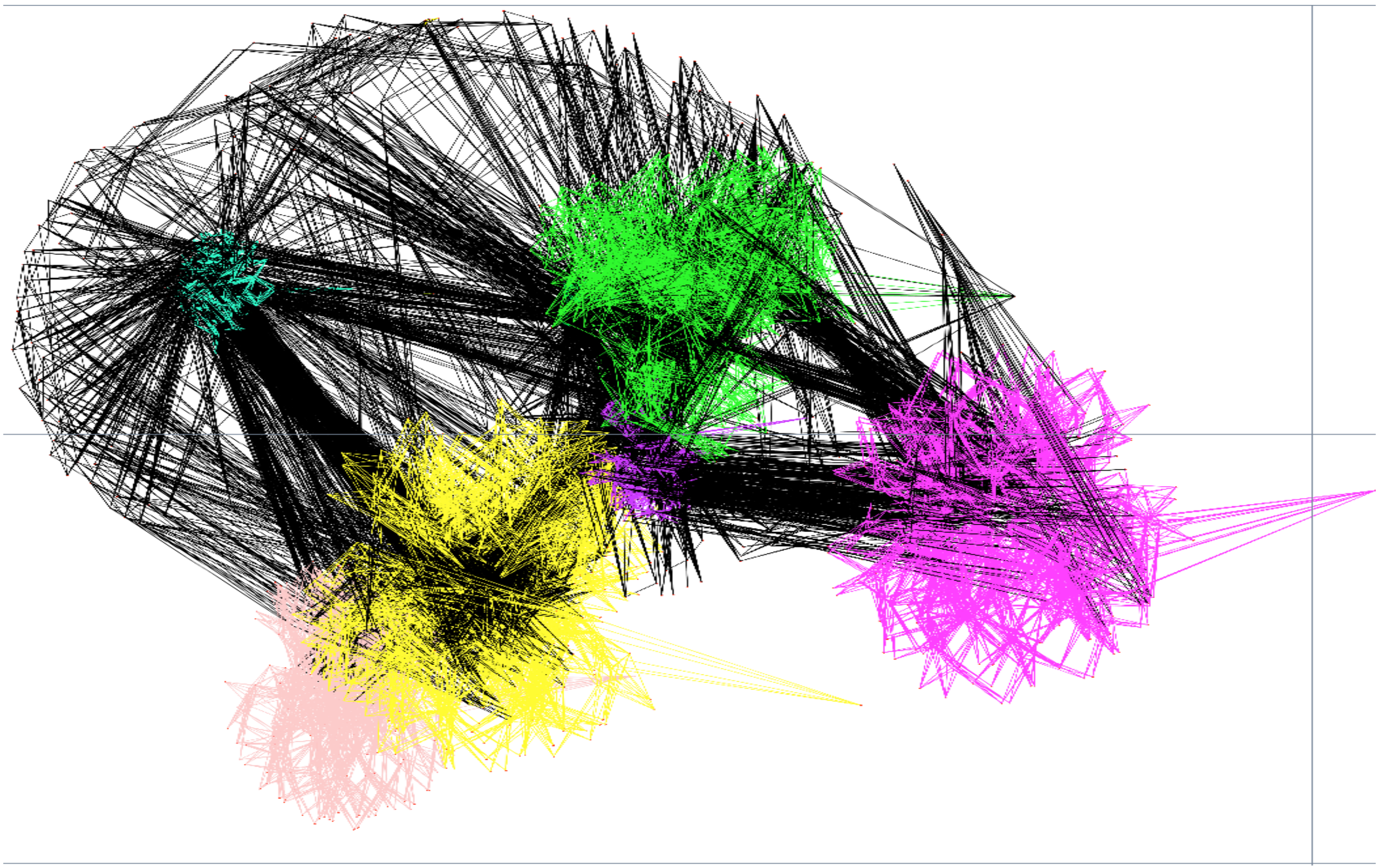
- It is characterized using a metric called Q (quality factor) that ranges from 0 to 1

- Informally, if a graph has lots of small clusters that are weakly connected (easily separable) to each other then such a graph is said to have high Q

- If a graph looks like a “giant hairy ball” then it has low Q

Community Structure in Graphs

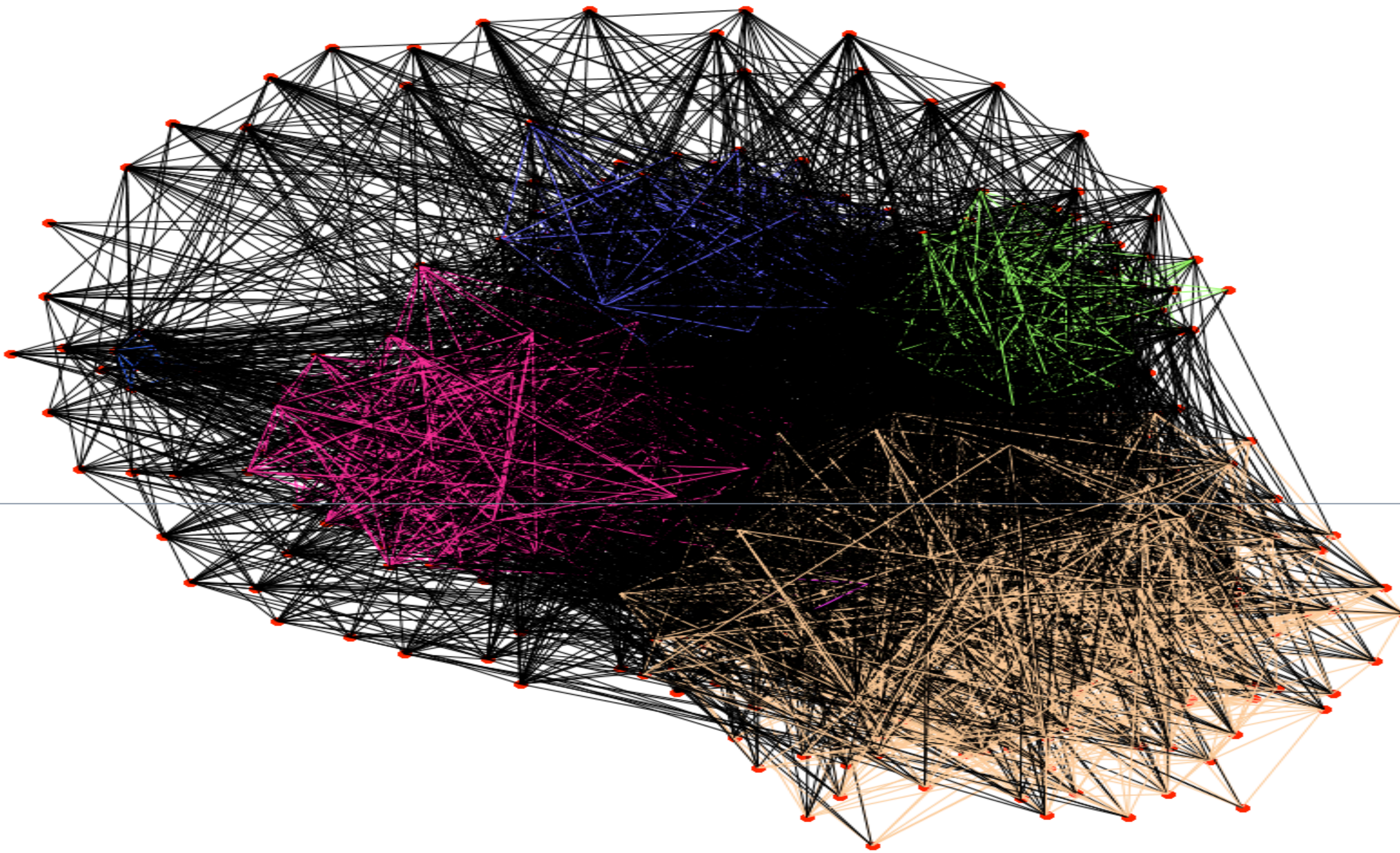
Variable-incidence Graph of Non-random Formula



SOURCE: mrpp example from SAT 2013 competition viewed using our SATGraf tool

Community Structure in Graphs

Variable-incidence Graph of Randomly-generated Formula



SOURCE: unif-k3-r4.267-v421-c1796-S4839562527790587617 **randomly-generated** example from SAT 2013 competition

Modularity (Q-factor) and Communities in Graphs

Community Structure in Graphs

- How to compute community structure?
- The decision version of the Q maximization problem is NP-complete [Brandes et al., 2006]
- Many efficient *approximate* algorithms proposed, e.g., [CNM04] and [0L13]
- We use the above two algorithms for our experiments
- Our results with both algorithms are similar

Community Structure and SAT Solver Performance
Our Results and Take-home Message

- A (partial) answer to question 1
  - A graph-theoretic characterization of SAT instances, as opposed to measuring the size of instances only in terms of number of variables and clauses
  - Industrial SAT instances have "good" community structure (also confirmed by previous work by Jordi Levy et al.)
  - Community structure of the graph of SAT instances strongly affect solver performance

- Result #1: Hard random instances have low Q ($0.05 \leq Q \leq 0.13$)
- Result #2: Number of communities and Q of SAT instances are more predictive of CDCL solver performance than other measures
- Result #3: Strong correlation between community structure and LBD (Literal Block Distance) in Glucose solver

Community Structure and Random Instances

Experiments #1: Hypothesis and Definitions

Hypothesis tested:

- Is there a range of Q values for randomly generated instances, that are hard for CDCL solvers; regardless of the number of clauses/variables
- Are randomly generated instances outside this range uniformly easy

Community Structure and Random Instances

Experiments #1: Setup

- Randomly generated 550,000 SAT instances for the experiment
  - Varied N_V between 500 and 2000 in increments of 100
  - Varied N_{cl} between 2000 and 10000 in increments of 1000
  - Varied target Q between 0 and 1 in increments of 0.01
  - Varied “Number of communities” between 20 and 400 in increments of 20
- Experiments using MiniSAT
  - Timeout of 900 seconds per run
  - Run solver on inputs in a random order
  - Average the running time over several runs

Community Structure and Random Instances

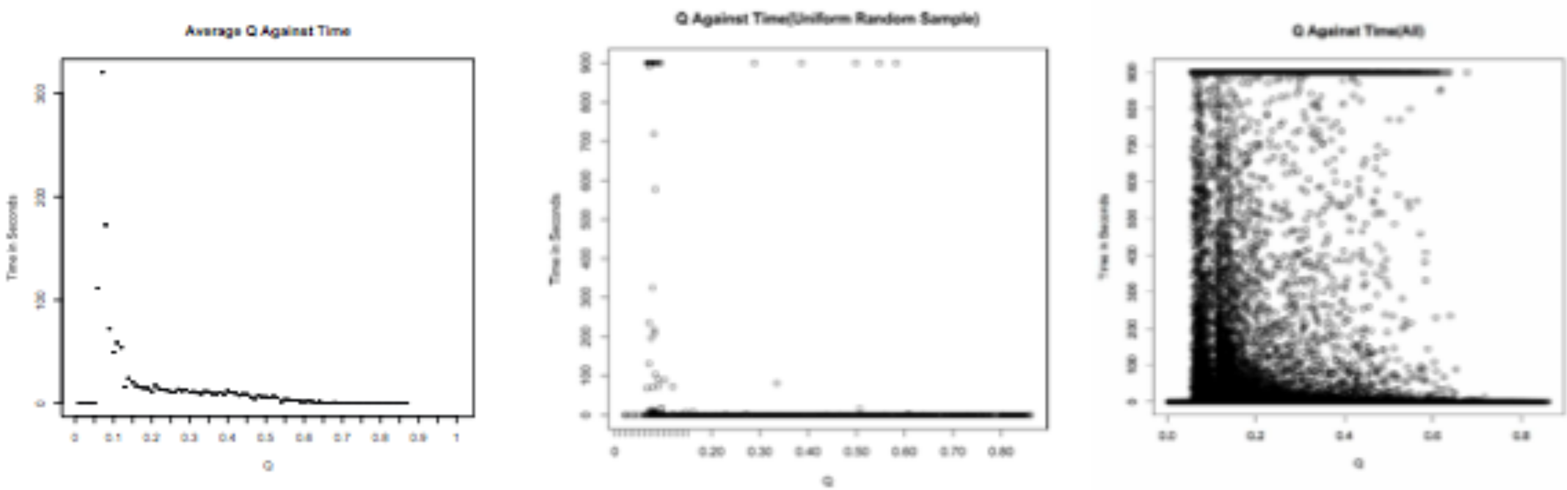
Experiments Performed (#1)

- Plotted Q against time
- Noticed significant increase in execution time when $0.05 \leq Q \leq 0.13$
- Also recomputed the results using a stratified sample
  - Used due to high number of instances within target range
  - Randomly sample the data taking 250 results from each 0.1 range of Q between 0 and 0.9
  - Almost the same result: $0.05 \leq Q \leq 0.12$

Community Structure and Random Instances

Experiments Performed (#1)

- Huge increase in running time of randomly generated instances when $0.05 \leq Q \leq 0.13$



(a) Average Time (b) Stratified Sample (c) All instances

Community Structure and Industrial Instances

Experiments #2: Hypothesis and Definitions

Hypothesis tested:

- Are the community modularity and number of communities better correlated with the running time of CDCL solvers than traditional metrics

- Is the correlation better for industrial instances than randomly generated or hand crafted ones

Community Structure and Industrial Instances

Experiments #2: Hypothesis and Definitions

- Instances used
  - Approximately 800 instances from the SAT 2013 competition. For the remaining we couldn't compute community structure due to resource constraints
  - Using OL algorithm to compute community structure for the 800 instances. Much faster and more scalable
- All experimental results are for Minipure
  - Obtained from the SAT 2013 competition website
- Used statistical tool R to perform standard linear regression

Community Structure and Industrial Instances

Experiments Performed (#2)

- Performed linear regression on the solver running time twice
  - Once with community structure metrics (and variables/clauses)
  - Once without
- Compared the adjusted R^2 (variability) from both experiments
  - Variability measures how good the models predicted results are, compared with the actual results
  - Varies from 0 to 1
- The lower the variability (higher the R^2) the more predictive the model

Community Structure and Industrial Instances

Experiments Performed (#2)

- Timeouts included
  - A large portion (Approximately 60%) of the instances timedout
  - Not ideal, but without them there isn't enough data
- log(time) used
  - Timeouts
  - Wide distribution between instances that finished and timedout
- Data standardized to have mean = 0 and standard deviation = 1
  - Standard practice when regressors are in different scales.

Community Structure and Industrial Instances

Experiments Performed (#2)

- Model #1 - $R^2 \sim 0.5$
  - log(time) ~ |CL| * |V| * Q * |CO| * QCOR * CLVR
  - * denotes interaction terms between factors
  - |CL| = number of clauses
  - |V| = number of variables
  - |CO| = number of communities
  - QCOR = ratio of Q to communities
  - CLVR = ratio of clauses to variables

- Model #2 - $R^2 \sim 0.33$
  - log(time) ~ |CL| * |V| * CLVR

Community Structure and Industrial Instances

Experiments #2: Results and Interpretation

- The regressions show us that the model with the community structure metrics is a better predictor of running time than traditional metrics, i.e. number of clauses/variables.

Factor	Estimate	Std. Error	t value	$Pr(>\|t\|)$	Sig
$\|CO\|$	-1.237e+00	3.202e-01	-3.864	0.000121	***
$\|CL\| \odot Q \odot QCOR$	-4.226e+02	1.207e+02	-3.500	0.000492	***
$\|CL\| \odot Q$	-2.137e+02	6.136e+01	-3.483	0.000523	***
$\|CL\| \odot Q \odot \|CO\| \odot QCOR \odot VCLR$	-1.177e+03	3.461e+02	-3.402	0.000702	***
$\|CL\| \odot Q \odot \|CO\|$	-6.024e+02	1.774e+02	-3.396	0.000719	***
$Q \odot QCOR$	3.415e+02	1.023e+02	3.339	0.000881	***
Q	1.726e+02	5.200e+01	3.318	0.000947	***
$Q \odot \|CO\| \odot QCOR$	9.451e+02	2.927e+02	3.229	0.001292	**

Literal Block Distance (LBD) and Communities

Experiment #3: Hypothesis and Definitions

Hypothesis tested

- The number of communities in a conflict clause correlates strongly with its LBD measure

What is LBD? (Glucose solver [AS09])

- LBD measure M of a learnt clause C is a rank based on the number N of distinct decision levels the vars in C belong to
- The lower the value of N the better the clause C is
- LBD is a powerful measure of the utility of a conflict clause

Literal Block Distance (LBD) and Communities

Experiment #3: Hypothesis and Definitions

- LBD and Clause deletion
  - Integral to the efficiency of modern solvers
  - Without clause deletion, conflict clause production quickly consumes available memory
  - LBD is a useful in determining which clauses to delete
- Which clauses to delete? LBD to the rescue
  - Periodically delete conflict clauses with bad LBD rank
  - As we will see, clauses with bad LBD rank are shared by many communities

Literal Block Distance (LBD) and Communities

Experiment #3: Intuition

- The number of communities in a conflict clause
  - The number of communities N in a conflict clause C is the number of distinct communities the variables in C belong to
- Intuition behind the hypothesis
  - High quality conflict clauses tend to span very few communities, i.e. N is small
  - High quality conflict clauses are likely to cause more propagation per decision variable, and hence are likely to have low LBD
  - LBD picks out high quality conflict clauses

Literal Block Distance (LBD) and Communities

Experiment #3: Setup

- Instances considered
  - 189 SAT 2013 industrial category instances out of 300
  - We were only able to compute communities for these 189
  - The rest caused memory-out errors
- Step 1
  - For each of the 189 instances, compute:
    - Community structure
    - The number of communities a learnt clause spans
    - LBD of every learnt clause (only for the first 20,000 due to resource constraints)

Literal Block Distance (LBD) and Communities

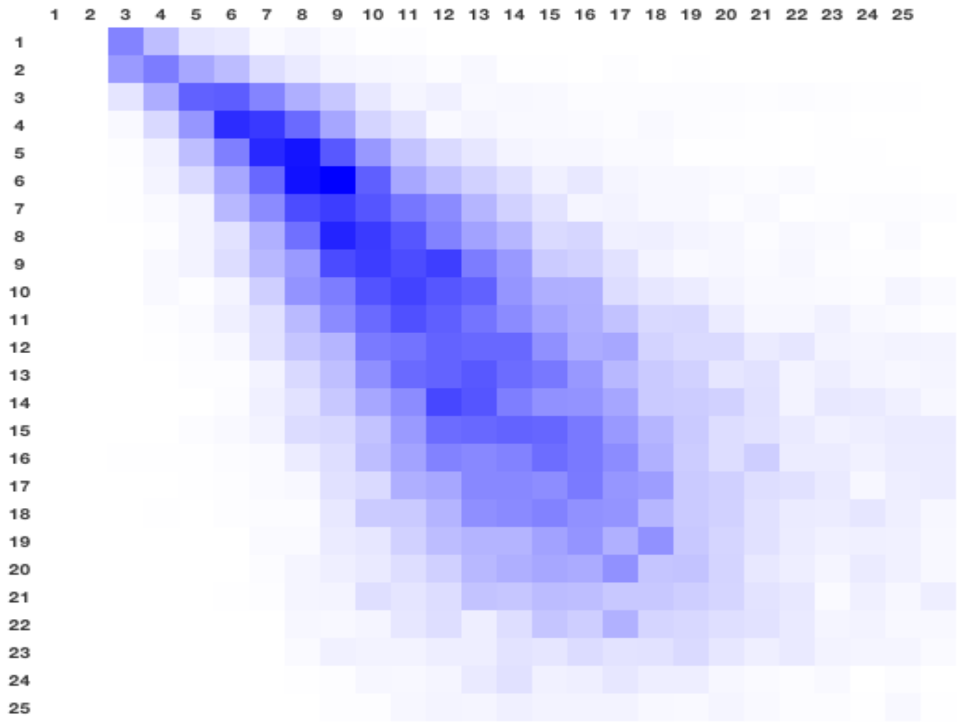
Experiments Performed (#3)

- Step 2
  - LBD of every learnt clause considered, was correlated with the number of communities it spans
  - Thousands of data points over the 189 instances
- Correlate LBD and number of communities using heatmaps
  - Heatmap of LBD and communities of learnt clauses
  - Difficult to correlate thousands of data points over hundreds of instances
  - One heatmap per SAT instance

Literal Block Distance (LBD) and Communities

Experiments #3: Results and Interpretation

- Result
  - Most industrial instances have a very strong correlation between LBD and communities



Impact of Community Structure and Solver Running Time

Scope for Improvement

- Consider different regression techniques
  - The non-normality of the data stops us from estimating confidence intervals
- Try experiments on more solvers
  - Glucose, MiniSAT and Minipure were the solvers we considered so far
- Compare different random generation techniques, and different graph representation for SAT instances
- Make the community-structure based model more robust by adding other features of SAT instances
- Compare against other models proposed based on backdoors and graph-width
- Construct a predictive model

The Laws of SAT Solving

We Provided an Answer to Question 1

- We break the problem statement down to smaller sub-problems

1. On which class of instances do SAT solvers perform well? I.e., a precise mathematical characterization of instances on which solvers work well

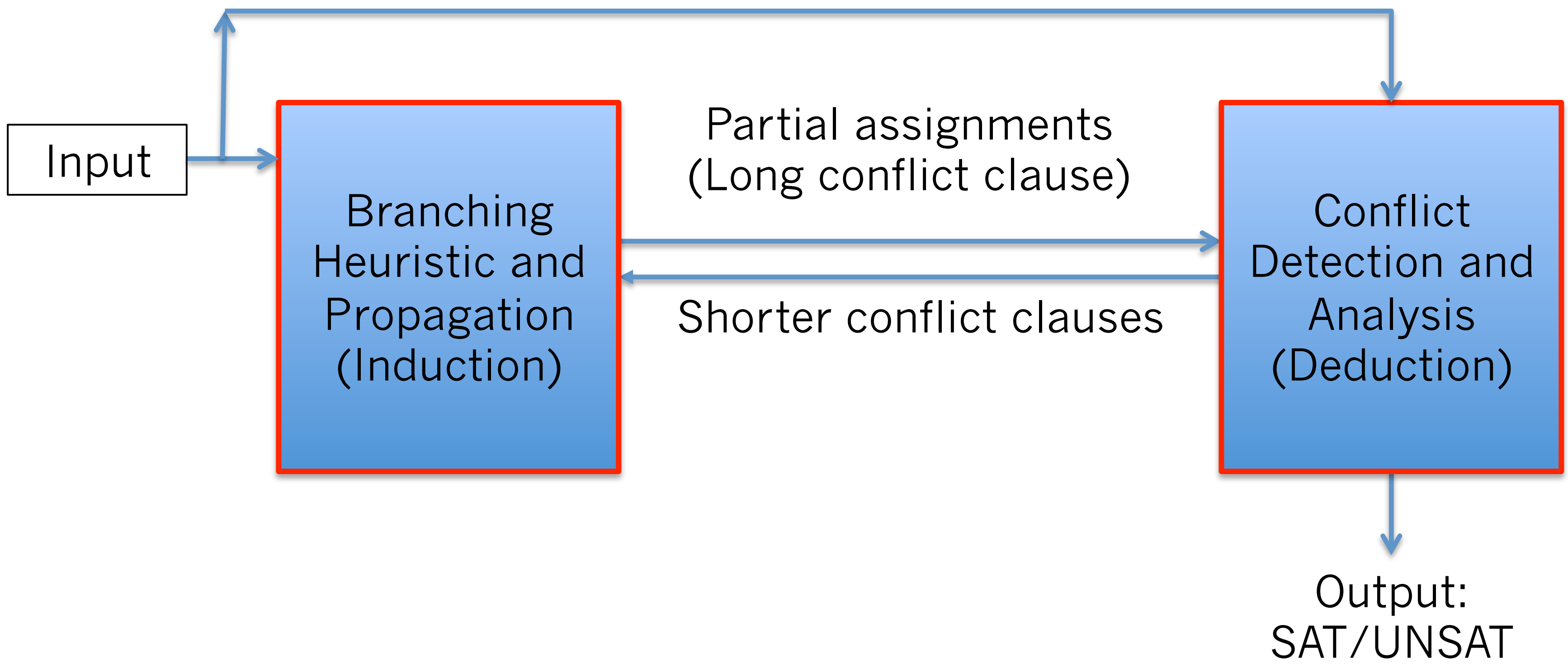
2. An abstract algorithmic description of SAT solvers

3. A complexity-theoretic analysis that provides meaningful asymptotic bounds

In this talk, I focus on Question 1, and briefly touch upon some potential answers for Question 2.

A Model for CDCL Solvers

CDCL Solvers: Induction and Deduction with Feedback



Community Structure and SAT Solver Performance

Conclusions and Take-home Message

- A (partial) answer to question 1
  - A graph-theoretic characterization of SAT instances, as opposed to measuring the size of instances only in terms of number of variables and clauses
  - Industrial SAT instances have “good” community structure (also confirmed by previous work by Jordi Levy et al.)
  - Community structure of the graph of SAT instances strongly affect solver performance

- Result #1: Hard random instances have low Q ($0.05 \leq Q \leq 0.13$)
- Result #2: Number of communities and Q of SAT instances are more predictive of CDCL solver performance than other measures (for the Minipure solver)
- Result #3: Strong correlation between community structure and LBD (Literal Block Distance) in Glucose solver

Community Structure in Graphs

Questions