PhD-iFM'17, Sept. 19th, Torino (IT)

ON THE COMMUNITY STRUCTURE OF SAT-BMC INSTANCES

Xavier Gillard, Charles Pecheur
Université Catholique de Louvain (BE)

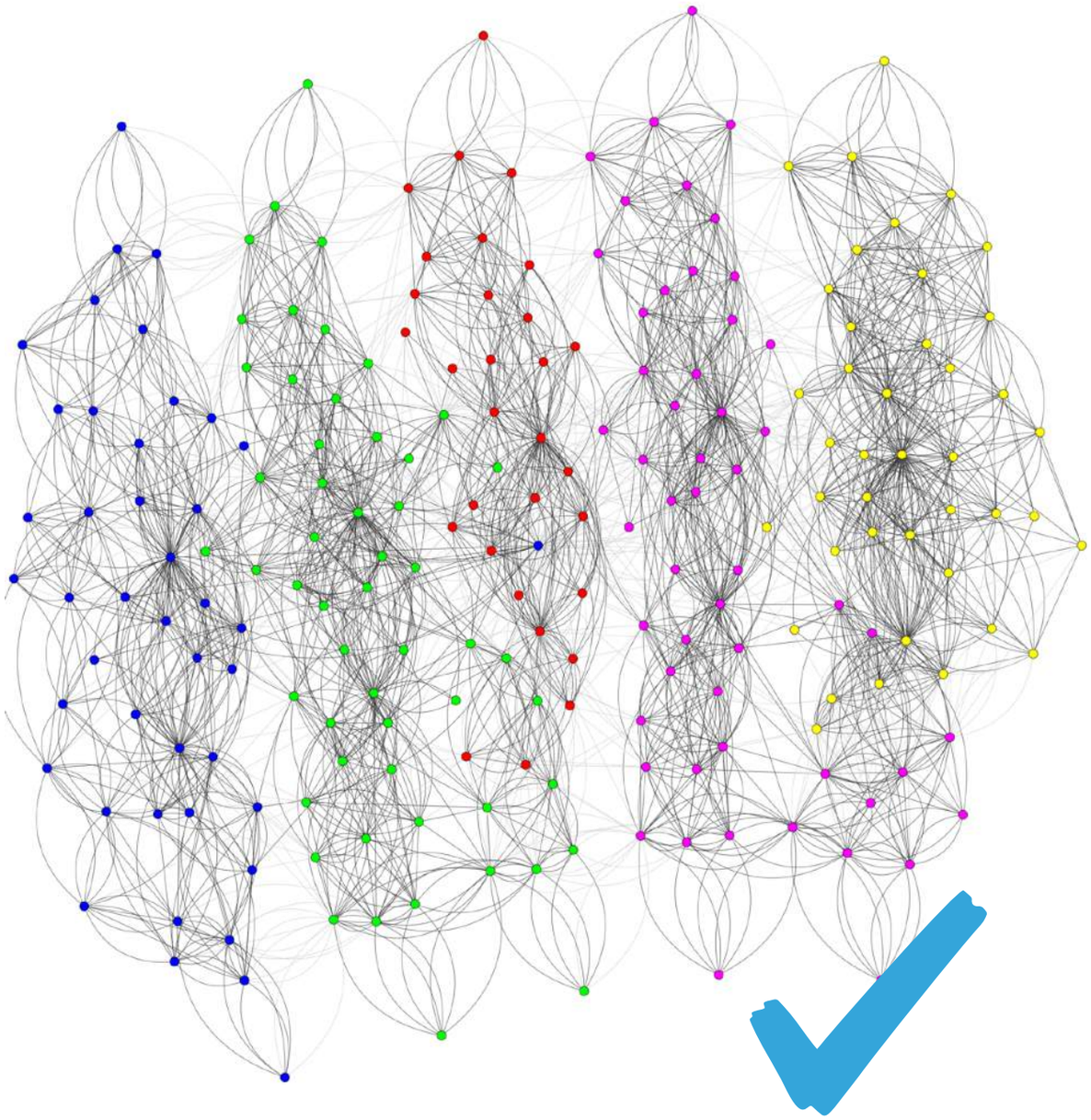
BACKGROUND

PhD-iFM'17, Sept. 19th, Torino (IT)

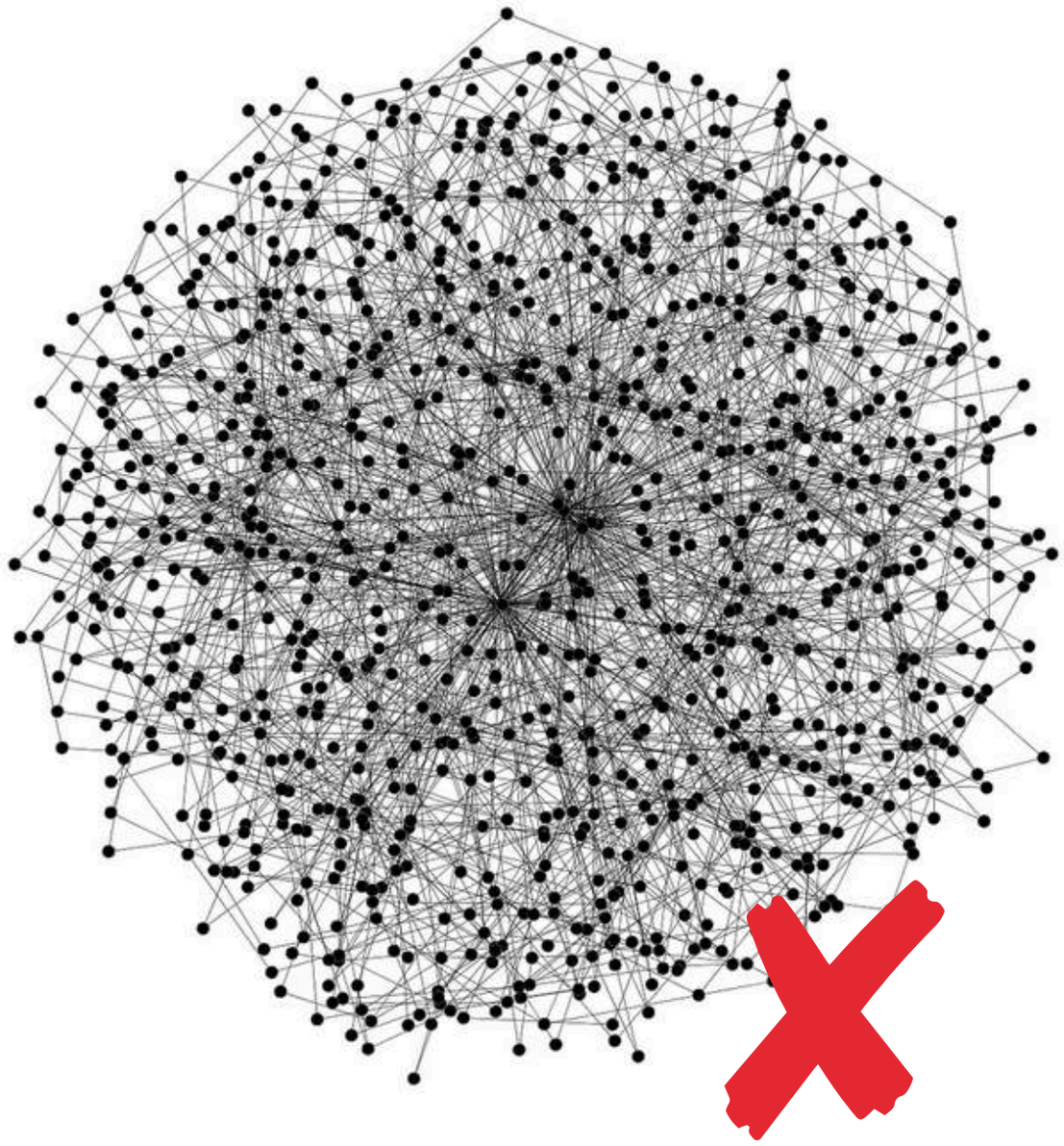
ON THE COMMUNITY STRUCTURE OF SAT-BMC INSTANCES

Xavier Gillard, Charles Pecheur
Université Catholique de Louvain (BE)

« GOOD » COMMUNITY STRUCTURE



Modularity = 0.6

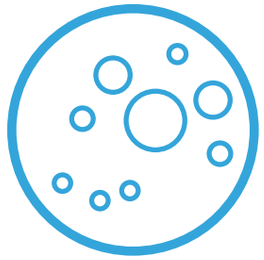


Modularity = ~0

COMMUNITY

Intuitively, a **community** is a cluster of nodes

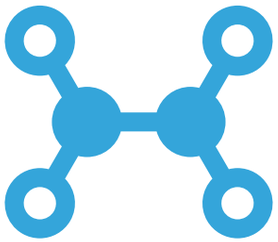
Densely connected with one another



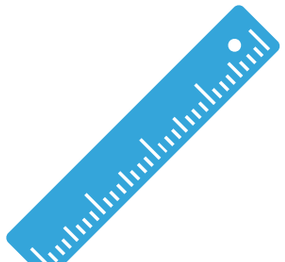
Sparsely connected with the rest of the graph

COMMUNITY STRUCTURE

The **community structure** of a graph is

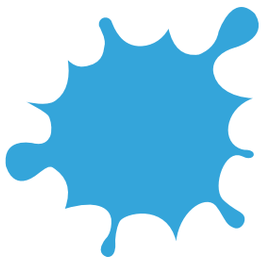


The propensity of a graph
to be decomposed in
almost disjoint subgraphs
(communities)



Measured using the q-score
aka **modularity metric**

Dire que le Q score est une metric qui parmet d'évaluer à quel point les densités interne / externe sont différentes de ce qu'on aurait observé sur un graphe random equivalent



Estimated w/ efficient
algorithms like i.e. the
Louvain Method

BOUNDED MODEL CHECKING

Bounded Model Checking, aka SAT-BMC is

Symbolic **Verfication** technique

Linear **Temporal** Logic

Bug Finding mode

Fully automatic thanks to **SAT** reduction

BOUNDED MODEL CHECKING: EXAMPLE, TIME 0

$$\phi = \mathbf{G}\ \neg crash$$

BOUNDED MODEL CHECKING: EXAMPLE, TIME 1

$$\phi = \mathbf{G}\ \neg crash$$

BOUNDED MODEL CHECKING: EXAMPLE, TIME 2

$$\phi = \mathbf{G}\ \neg crash$$

BOUNDED MODEL CHECKING: SIZE OF THE SAT PROBLEM

Time 0 Time 1 Time 2

SAT SAT SAT

MODERN SAT SOLVERS

One of the **biggest success** in computer science since the early 2000's

Reduction to SAT **often outperforms** ad hoc techniques

Solve **computationally hard** problems that were previously thought to be infeasible

MODERN SAT SOLVERS

Modern SAT solvers implement **Conflict Driven Clause Learning** (CDCL)

Build upon **DPLL**

Learn new clauses from their errors

Use **heuristics** to maximize the information gained from learning (VSIDS)

MOTIVATION

WE KNOW CDCL SOLVERS WORK, BUT WE DON'T KNOW WHY THEY WORK...

FACT

CDCL solvers perform very well on industrial instances and not so good on random problems

INTUITION

CDCL solvers are able to exploit the underlying **problem structure**

What does it mean ? What structure ?

COMMUNITY STRUCTURE IS THE BEST DEFINITION SO FAR…

Good for industrial problems, Bad for random instances [Ansotegui12]
Correlates with CDCL runtime [Newsham14]
VSIDS picks high-centrality bridge variables [Liang15]
Community structure reveals non-trivial structure existing in the CNF [BaudBerthier17]

[Mull16] proved that CDCL performs poorly on some pseudo-industrial instances despite their having a good community structure.

WHAT IS SO SPECIAL ABOUT THE COMMUNITY STRUCTURE OF INDUSTRIAL SAT INSTANCES ?

WHAT DOES IT MEAN ?

CONTRIBUTION

TOOL TO ANALYZE THE COMMUNITY STRUCTURE OF BMC

+

pip install pynusmv-community

TOOL WORKFLOW

1. Start from SMV model
2. Generate SAT problem
3. Analyze VIG structure
4. Semantic vertex reconciliation

FEATURES

Mining

- Patterns
- Sequences

Visualization

- VIG
- Cluster graph
- Timeline
- Clouds

Statistics

- #Communities
- Modularity

RAW DATA

- DIMACS file
- Communities
- Semantic mapping
- FCA

METHODOLOGY

QUALITATIVE APPROACH

MODEL OF A RAILWAY INTERLOCKING (NAMECHE, BELGIUM)

MODELS FROM THE NUSMV EXAMPLES COLLECTION

OBSERVATIONS ABOUT THE COMMUNITIES

Alternating Bit Protocol
(NuSMV, ABP4, BIERE)

Variable		0	1	2	3	4	5	6	7	8		10
???												
_process_selector_												
r2s_in.tag												
r2s_out.tag												
receiver.abp												
receiver.data												
receiver.state												
s2r_in.data												
s2r_in.tag												
s2r_out.data												
s2r_out.tag												
sender.abp												
sender.data												
sender.state												

COMMUNITIES TO WHICH (AT LEAST ONE BIT OF)
THE VARIABLE BELONG

OBSERVATIONS ABOUT THE COMMUNITIES

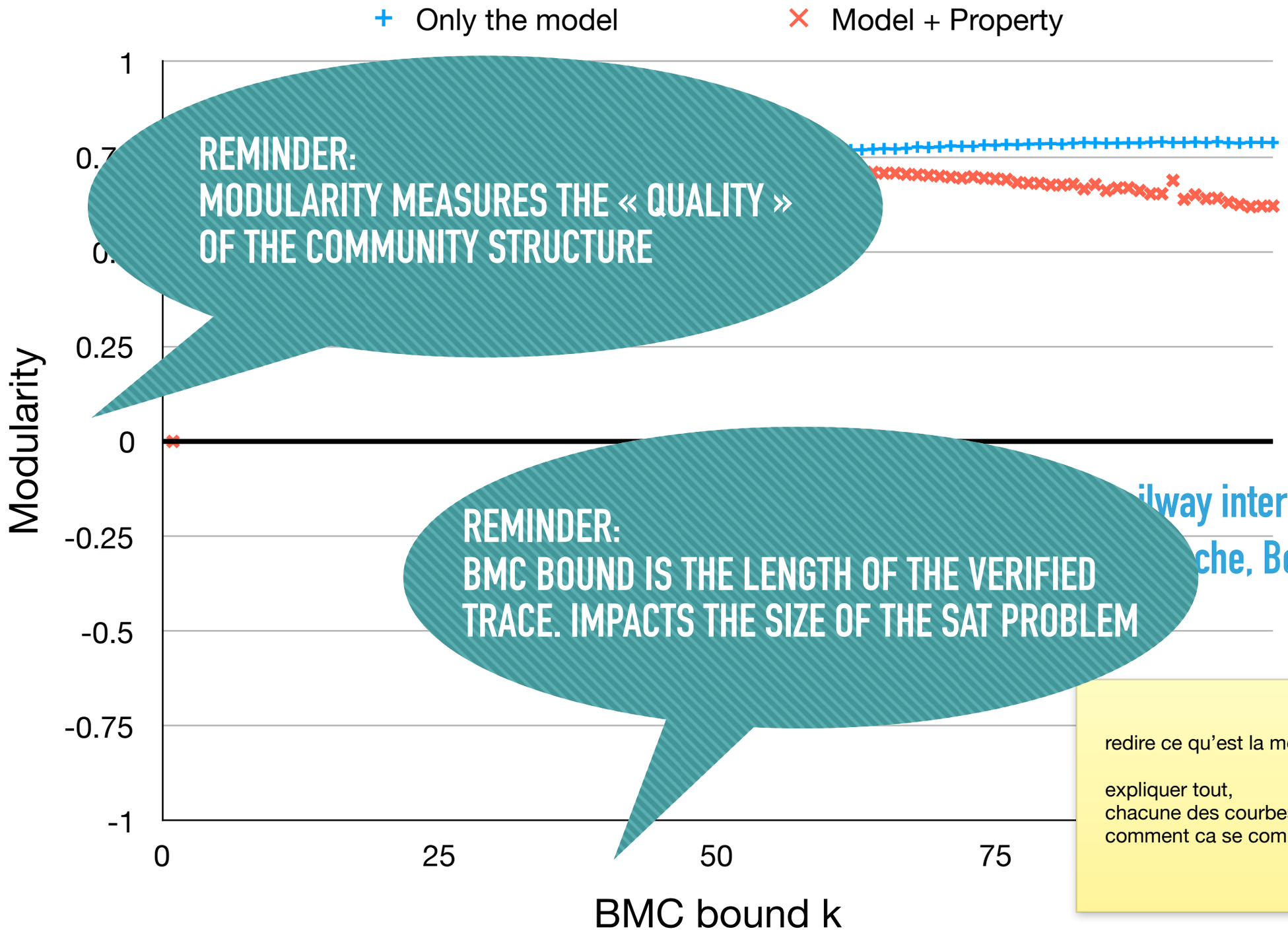
Alternating Bit Protocol
(NuSMV, ABP4, BIERE)

Variable	-1	0	1	2	3	4	5	6	7	8	9	10
???												
_process_selector_	11	11	2	2	3	3	10	10	9	9		
r2s_in.tag	4	4	5	5	6	6	7		8	8	12	
r2s_out.tag	11	4	4	5	5	6	6		7	8	8	
receiver.abp	4	4	5	5	6	6	7		8	8	8	
receiver.data	4,12	4,					2,6	7	12,7	8,12	12	
receiver.state	4	4					7		8	8	12	
s2r_in.data	11,12	11,12	2,12	2,12	3,12	3,12	10,12	12	9,12	9,12	12	
s2r_in.tag	11	11	2	2	3	3	10		9	9	12	
s2r_out.data	4,12	4,12	12,5	12,5	12,6	12,6	12,7	7	8,12	8,12	12	
s2r_out.tag	4	11	11	2	2	3	3	10	10	9	9	
sender.abp	11	11	2	2	3	3	10	10	9	9	12	
sender.data	11,12	11,12	2,12	2,12	3,12	3,12	10,12	10,12	9,12	9,12	12	
sender.state	11	11	2	2	3	3	10	10	9	9	9,12	

INCREASES W/ BOUND

SYNCHRONIZATION ?

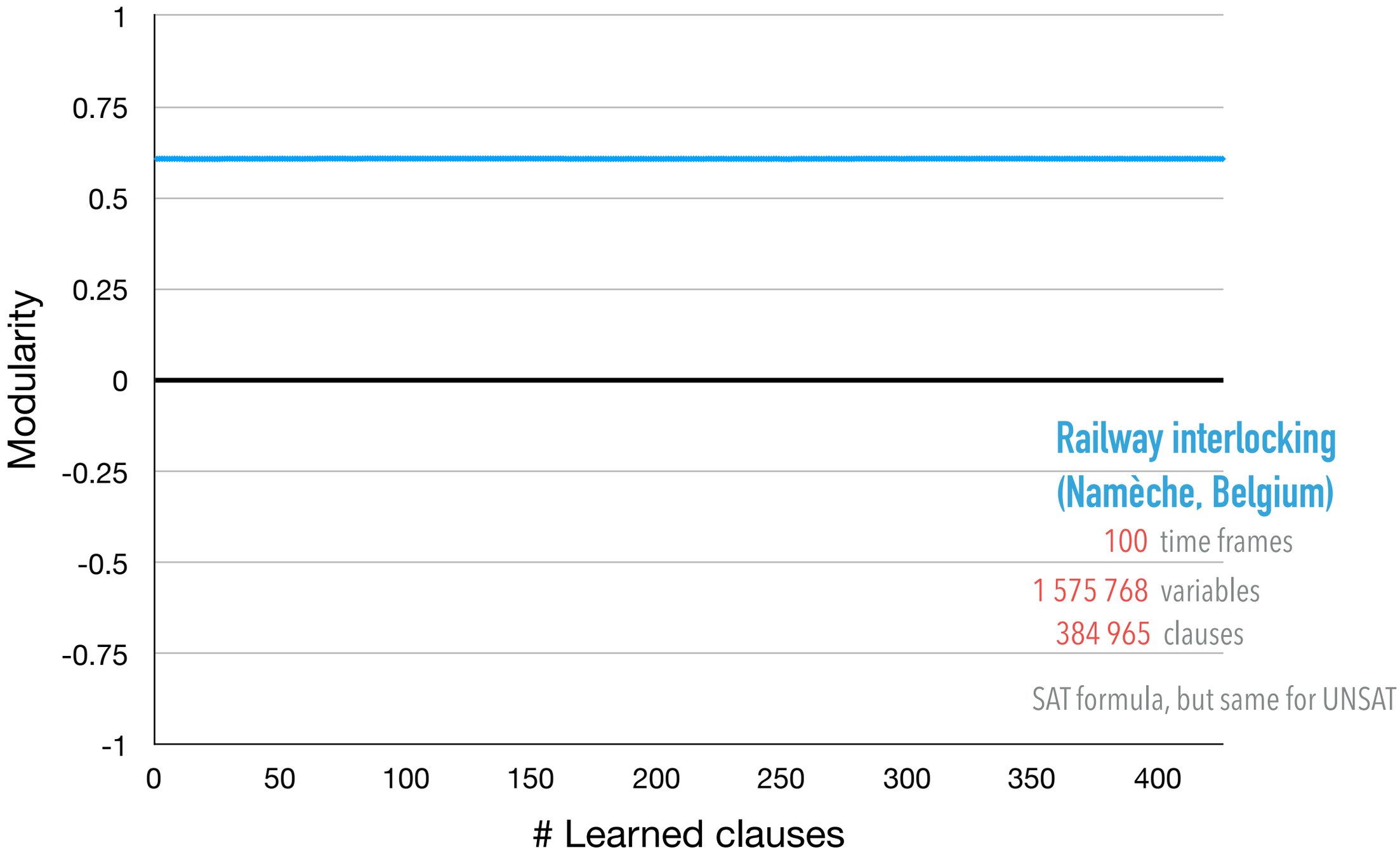
EVOLUTION OF THE MODULARITY WRT BMC BOUND



redire ce qu'est la modularité

expliquer tout,
chacune des courbes
comment ca se comporte

EVOLUTION OF THE MODULARITY WRT LEARNED CLAUSES



FUTURE WORK

CONFIRM HYPOTHESIS ABOUT THE SLIDING PATTERNS

INVESTIGATE WHETHER BRIDGE VARIABLES UNVEIL MODEL CONCEPTS

ANALYZE THE MEANING OF LEARNED CLAUSES

CONCLUSIONS

- Analysis of the community structure reveals **patterns sliding** along the time line
- Learning clauses **does not improve modularity**

REFERENCES

BIBLIOGRAPHY

[Ansotegui12] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy.
The community structure of SAT formulas.
In International Conference on Theory and Applications of Satisfiability Testing,
pages 410–423. Springer, 2012.

[Newsham14] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, Laurent Simon
Impact of community structure on SAT solver performance
In International Conference on Theory and Applications of Satisfiability Testing,
pages 252–268. Springer, 2014.

[Liang15] Liang, J. H., Ganesh, V., Zulkoski, E., Zaman, A., & Czarnecki, K
Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers
In Haifa Verification Conference (pp. 225-241). Springer, 2015

[Mull16] Nathan Mull, Daniel J Fremont, and Sanjit A Seshia
On the hardness of SAT with community structure
In International Conference on Theory and Applications of Satisfiability Testing,
pages 141-159, Springer 2016

[BaudBerthier17] Guillaume Baud-Berthier, Jesús Giráldez-Cru, Laurent Simon
On the Community Structure of Bounded Model Checking SAT Problems.
In International Conference on Theory and Applications of Satisfiability Testing,
pages 65-82. Springer, 2017

BIBLIOGRAPHY

[Newman06] M.E.J. Newman
Modularity and community structure in networks
In Proceedings of the national academy of sciences, 103(23), 8577-8582.

[Blondel08] Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre
Fast unfolding of communities in large networks
In Journal of statistical mechanics: theory and experiment, 2008(10), P10008.

[Busard13] Simon Busard, Charles Pecheur
PyNuSMV: NuSMV as a python library
In NASA Formal Methods Symposium, 2013, Springer.

[Gillard16] Xavier Gillard
Adding SAT-based model checking to the pynusmv framework
M.Sc Thesis, Université Catholique de Louvain, 2016

BACKUP SLIDES

BOUNDED MODEL CHECKING: CONCRETELY

Initial State

Transition Relation

Straight Path

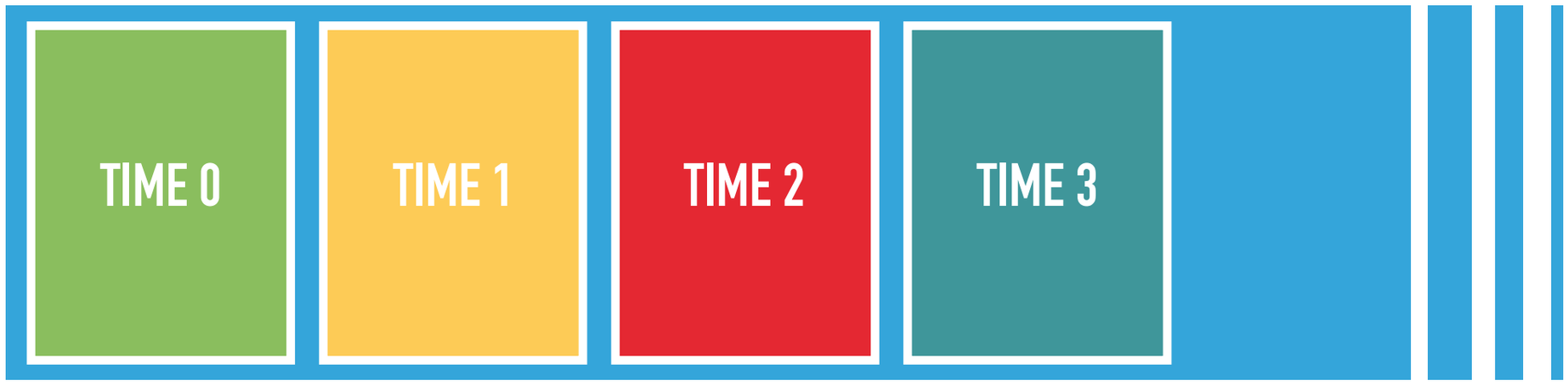
Looping Path

$$\underbrace{\left(\llbracket S_0 \rrbracket \wedge \bigwedge_{i=0}^{k-1} \llbracket T \rrbracket_k^i\right)}_{\text{The automaton formula } \llbracket K \rrbracket_k} \wedge \underbrace{\left(\llbracket \neg\phi \rrbracket_k^0 \vee \bigvee_{l=0}^{k} \left({}_lL_k \wedge {}_l\llbracket \neg\phi \rrbracket_k^0\right)\right)}_{\neg\phi \text{ bounded semantics}}$$

Any Loop

Loop condition

BOUNDED MODEL CHECKING: CONCRETELY



Trains qui se roulent dessus

+ phi1 > phi2 > ph3

NOT FEW EDGES BETWEEN THE COMMUNITIES BUT FEWER THAN EXPECTED

M.E.J Newman

MODULARITY – BI PARTITION

[Newman06]

PURELY CONVENTION

NOT JUST A TOKEN OF RANDOMNESS

OF THE

DEGREE OF THE VERTICES

(HERE DEGREE OF v_i)

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1)$$

$$B_{ij} = \left(A_{ij} - \frac{k_i k_j}{2m} \right)$$

TOTAL # EDGES IN

$s_i = 1$ IF i BELONGS TO GROUP A

-1 IF i BELONGS TO GROUP B

MODULARITY – MULTI PARTITION (= REPEATED BI PARTITION)

[Newman06]

Bridges cannot simply be forgotten between iterations

$$\underbrace{\Delta Q = \frac{1}{4m}\left[\sum_{ij} B_{ij}(s_i s_j + 1) - \sum_{ij} B_{ij}\right]}_{\text{Contribution to the overall modularity}}$$

LOUVAIN METHOD – SLIGHT DEF. DIFFERENCES

[Blondel08]

$$Q = \frac{1}{2m} \left(\sum_{i,j} A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

LOUVAIN METHOD – SLIGHT DEF. DIFFERENCES

[Blondel08]

SUM OF THE WEIGHT OF THE LINKS INCIDENT TO NODE i

$$\Delta Q = \left[\frac{\sum in + k_{i,in}}{2m} - \left(\frac{\sum tot + k_i}{2m}\right)^2\right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

Contribution of moving i to the community C

LOUVAIN METHOD

Greedy

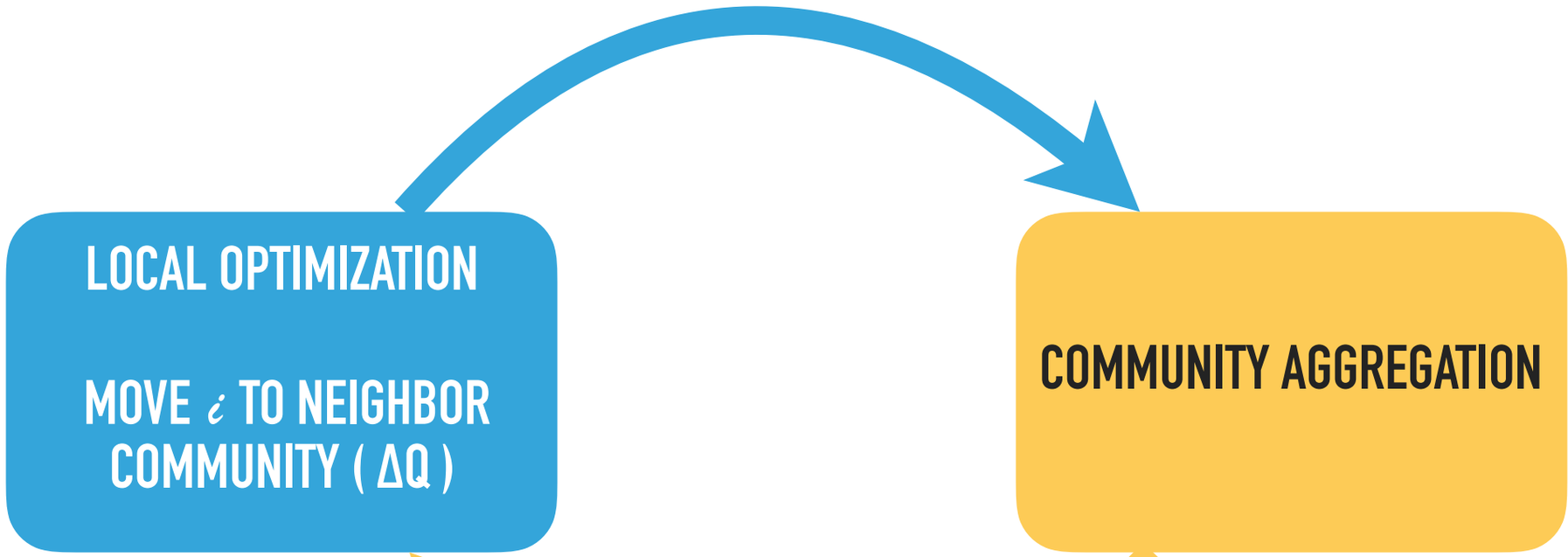
Bottom up

Meaningful intermediary results

No resolution limit

Simulation suggests
linear time complexity
on typical and sparse data

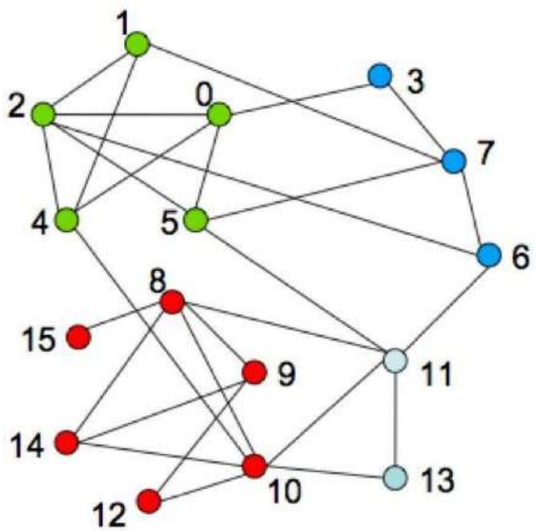
LOUVAIN METHOD – 2 PHASES



in this initial partition there are as many communities as there are nodes. Then, for each node i we consider the neighbours j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j. The node i is then placed in the community for which this gain is maximum (in case of a tie we use a breaking rule), but only if this gain is positive. If no positive gain is possible, i stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete. Let us insist on the fact that a node may be, and often is, considered several times. This first phase stops when a local maxima of the modularity is attained, i.e., when no individual move can improve the modularity. One should also note that the output of the algorithm depends on the order in which the nodes are considered. Preliminary results on several test cases seem to indicate that the ordering of the nodes does not

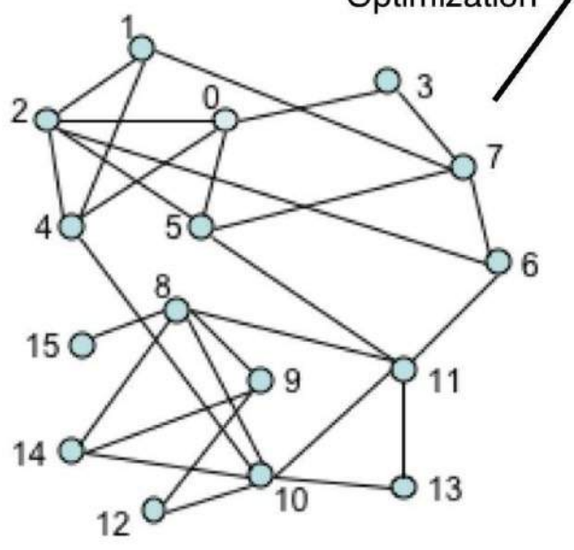
LOUVAIN METHOD – EXAMPLE

[Blondel08]

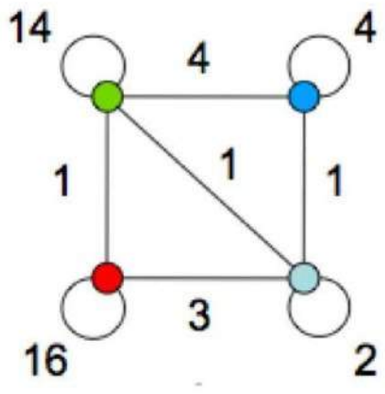


Modularity Optimization

Community Aggregation



1st pass



2nd pass