Consistencies and Boolean Satisfiability

Pavel Surynek
Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics
Charles University in Prague

http://ktiml.mff.cuni.cz/~surynek

Constraint Satisfaction Problem (CSP)

- Constraint satisfaction problem over the universe of elements $\mathbb{D}$ is a triple **(X,C,D)**
  - **X** – finite set of variables
  - **C** – finite set of constraints
  - **D** – is a function $D: X \rightarrow \mathcal{P}(\mathbb{D})$
  - each constraint $c \in C$ is a construct of the form $<(x_1^c, x_2^c, \dots, x_{k(c)}^c), R^c>$
    - $k(c)$ is arity of the constraint
    - $x_i^c \in X$ for I = 1,2,...,k(c) and $R^c \subseteq D(x_1^c) \times D(x_2^c) \times \dots \times D(x_{k(c)}^c)$

example: $\mathbb{D}=\{1,2,3\}$
$X=\{a,b,c\}$
C={<(a,b),"<">;
<(b,c),"=">}
$D(a)=D(b)=D(c)=\mathbb{D}$

- The task is to find **assignment of values to variables** from their domains such that all the constraints are satisfied
  - or decide that **no** such **valuation exists**

example: a=1, b=2, c=2

- Decision variant is an **NP-complete** problem

Boolean Satisfiability (SAT)

- A **Boolean formula** is given - variables can take either the value ***TRUE*** or ***FALSE***

example: $(\neg x \Rightarrow \neg y) \wedge (x \Rightarrow \neg y)$

- The task is to find **valuation of variables** such that the formula is **satisfied**
  - or decide that **no** such **valuation exists**

example: x = *TRUE*
y = *FALSE*

- Conjunctive normal form (**CNF**) - standard form of the input formula
  - **variables**: $x_1, x_2, x_3, \ldots$
  - **literals**: $x_1, \neg x_1, x_2, \neg x_2$, ... variable or its negation
  - **clauses**: $(x_1 \vee \neg x_2 \vee \neg x_3)$ ... disjunction of literals
  - **formula**: $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee \neg x_3)$ ... conjunction of clauses

example:
p cnf 3 2
1 -2 0
1 2 -3 0
...

- Clauses represent constraints that must be all satisfied (can be regarded as CSP) – SAT and CSP are mutually reducible

Motivation for Global Consistencies

- CSP paradigm provides many types of **local consistencies**
  - local inference is typically **too weak** for SAT
  - arc-consistency, path-consistency, i,j-consistency
    - insignificant gain in comparison with unit-propagation
    - expensive propagation with respect to the inference strength
- **Global** consistencies (global constraints)
  - provide strong global inference
    - often leads to significant simplification of the problem
  - application of **global consistencies** in SAT is quite rare
- Consistency based on **structural properties**
  - interpret SAT as a graph and find graph structures

Difficult Instances of SAT

- **Difficult** instances for **today's SAT** (more precisely for 2007's) solving systems
  - impossible to (heuristically) **guess** the solution
  - heuristics do not succeed ►► **search**
  - clause learning mechanism needs to learn for a long time
- Typical example: **unsatisfiable SAT** instances encoding Dirichlet's box principle (**Pigeon-hole principle**)
- **Satisfiable** case
  - **Valuation** of variables = certificate
  - **small witness** through which we can verify satisfiability
- **Unsatisfiable** case
  - no (small) witness (certificate) to guess
  - search/learning is necessary

Today's new **variable ordering heuristics** and **preprocessing** techniques can succeed on these types of instances.

Our Approach – conflict graphs

- **Input** - Boolean **formula in CNF**
- **Interpret** as a graph of conflicts
  - vertices = **literals**
  - edges = **conflicts** between literals
  - *example:* **x** and **¬x** are **in conflict** (cannot be satisfied together) ►► put an edge between corresponding vertices
- **Perform initial preprocessing**
  - Singleton unit propagation ►► **new conflicts**
  - Consistency based on **conflict graph**
- **Output** - equivalent **(simpler) formula** or the answer **"unsatisfiable"**

Initial Preprocessing – improve the graph

- Make the graph of conflicts **dense**
  - apply **singleton unit propagation**
  - discover **hidden** conflicts between literals
  - **denser** conflict graph **= better** for the subsequent step
- (Greedily) **find cliques** in the conflict graph
  - **at most one literal** from a clique can be satisfied
  - contribution of literal **x...c(x)** = number of clauses containing **x**
  - contribution of clique **C...c(C)** = $\max_{x \in C} c(x)$
  - $\boldsymbol{\sum_{C \in cliques} c(C) < \text{number of clauses}}$ (basic consistency check)
- All the cliques together do not contribute enough to satisfy the input formula ►► the input formula is **unsatisfiable**

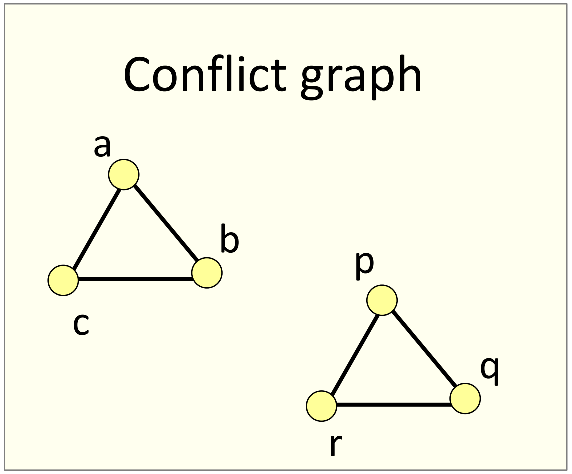
Clique Consistency – making projections

- Generalization of "$\sum_{C \in cliques} c(C) < \#clauses$"
- Choose a **sub-formula B = subset of clauses** and project the contribution counting on sub-formula
  - **contribution of literal x** to sub-formula **B** ...
    ...**c(x,B)** = number of clauses of **B** containing **x**
  - **contribution of clique C** to sub-formula **B** ...
    ...**c(C,B)** = $\max_{x \in C} c(x,B)$
  - when $\sum_{C \in cliques}$ **c(C,B)<number of clauses in B**
    ►► B is unsatisfiable ⇒ input formula is unsatisfiable
- **Singleton** approach...literal **x** is inconsistent
  - $\boldsymbol{\sum_{C \in cliques \not\ni x} c(C,B) < (\#clauses\ of\ B) - c(x,B)}$

 Pavel Surynek, 2011

Clique Consistency (example)

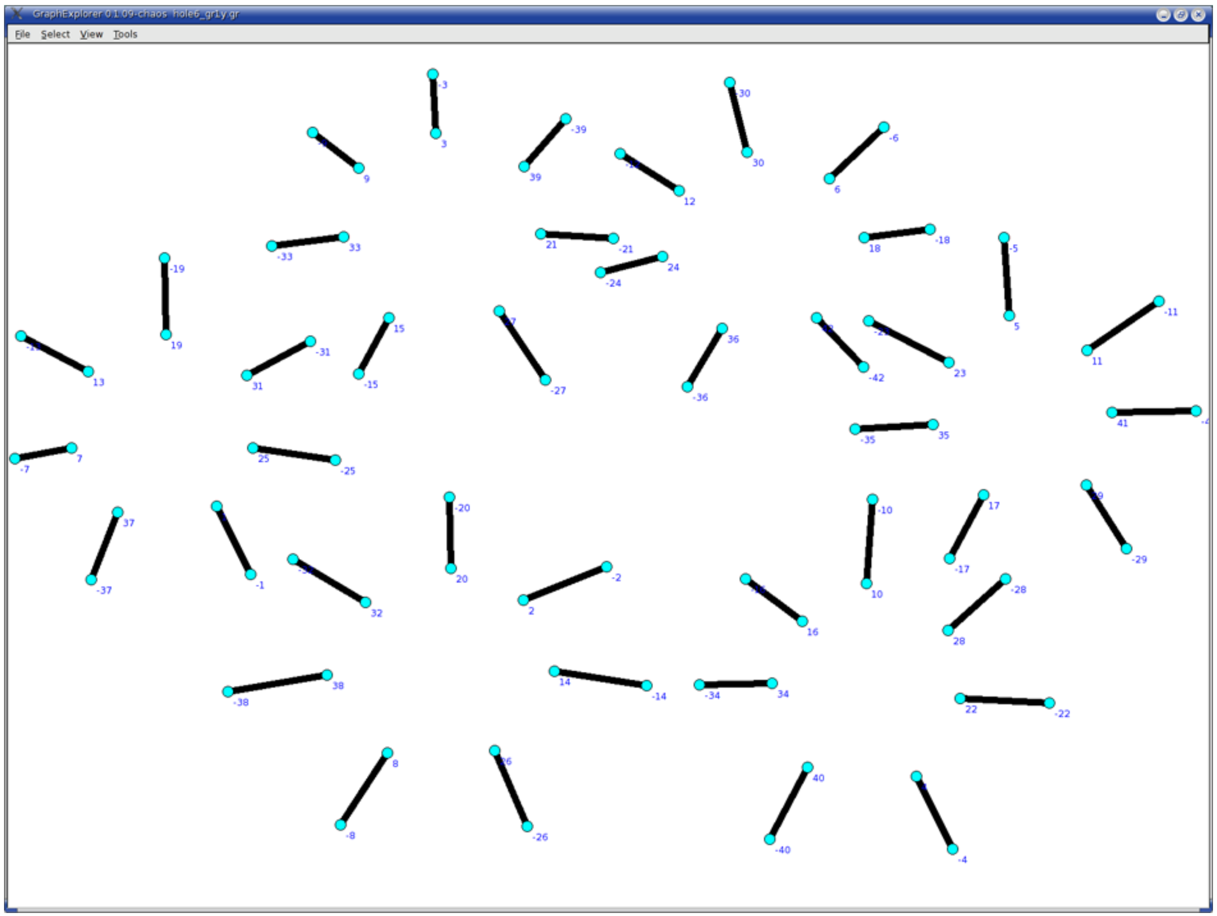
- Inconsistency (basic case – singleton approach is not applied):
  "$\sum_{C \in cliques} c(C,B) < \#clauses\ in\ B$"
  - **example:** clique $\mathbf{C_1}=\{\mathbf{a},\mathbf{b},\mathbf{c}\}$
    clique $\mathbf{C_2}=\{\mathbf{p},\mathbf{q},\mathbf{r}\}$
    - {**a**,**b**,**c**} are pair-wise conflicting
      {**p**,**q**,**r**} are pair-wise conflicting
  - sub-formula
    $\mathbf{B} = (\mathbf{a} \vee \mathbf{p})\ \&\ (\mathbf{b} \vee \mathbf{q})\ \&\ (\mathbf{c} \vee \mathbf{r})$
    $c(C_1,B)=1;\ c(C_2,B)=1$
  - $\sum_{C \in cliques} c(C,B) = 2$; #clauses in **B** = 3
- The original formula has **no** satisfying valuation.



Visualization (1)

using GraphExplorer software (Surynek, 2007-2010)

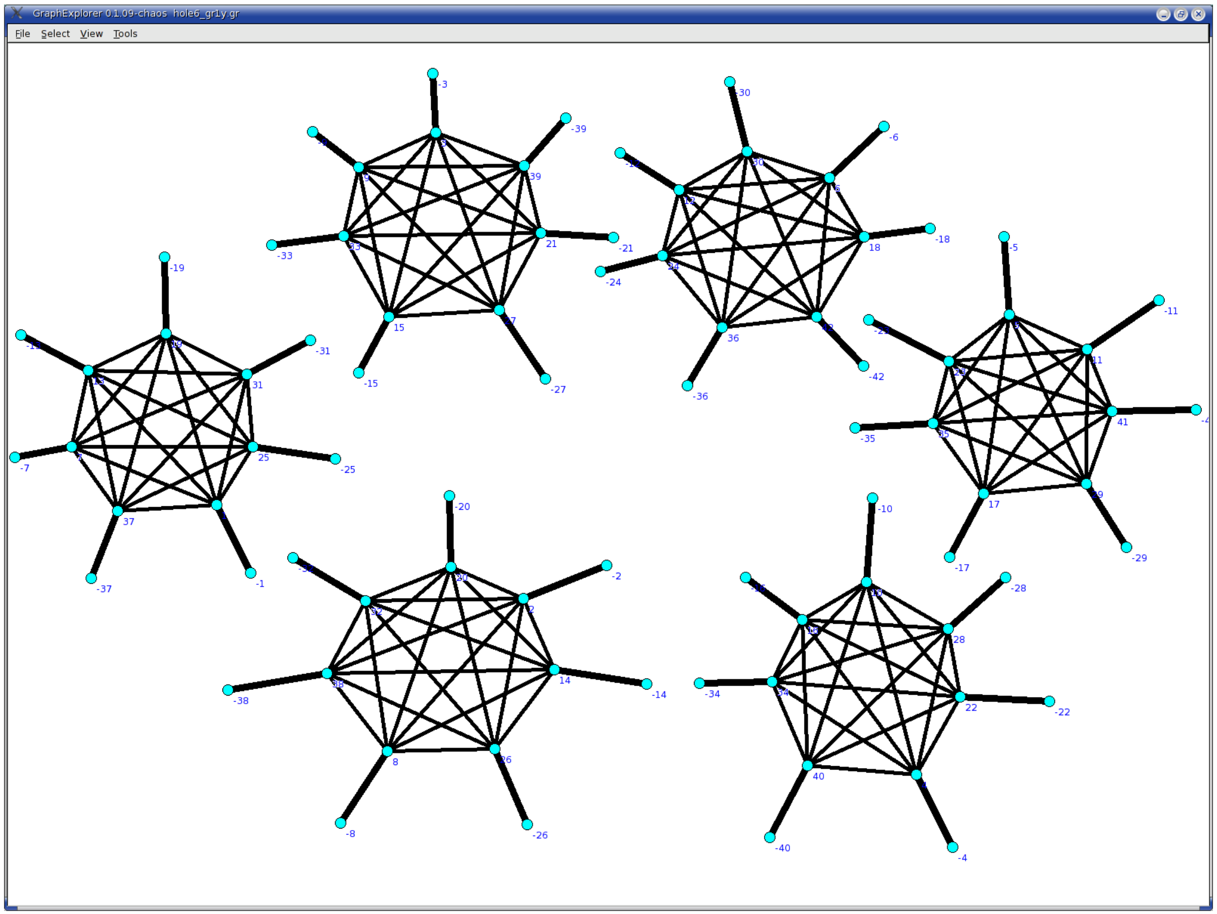
- **„Insert 7 pigeons into 6 holes“**



Visualization (2)

using GraphExplorer software (Surynek, 2007-2010)

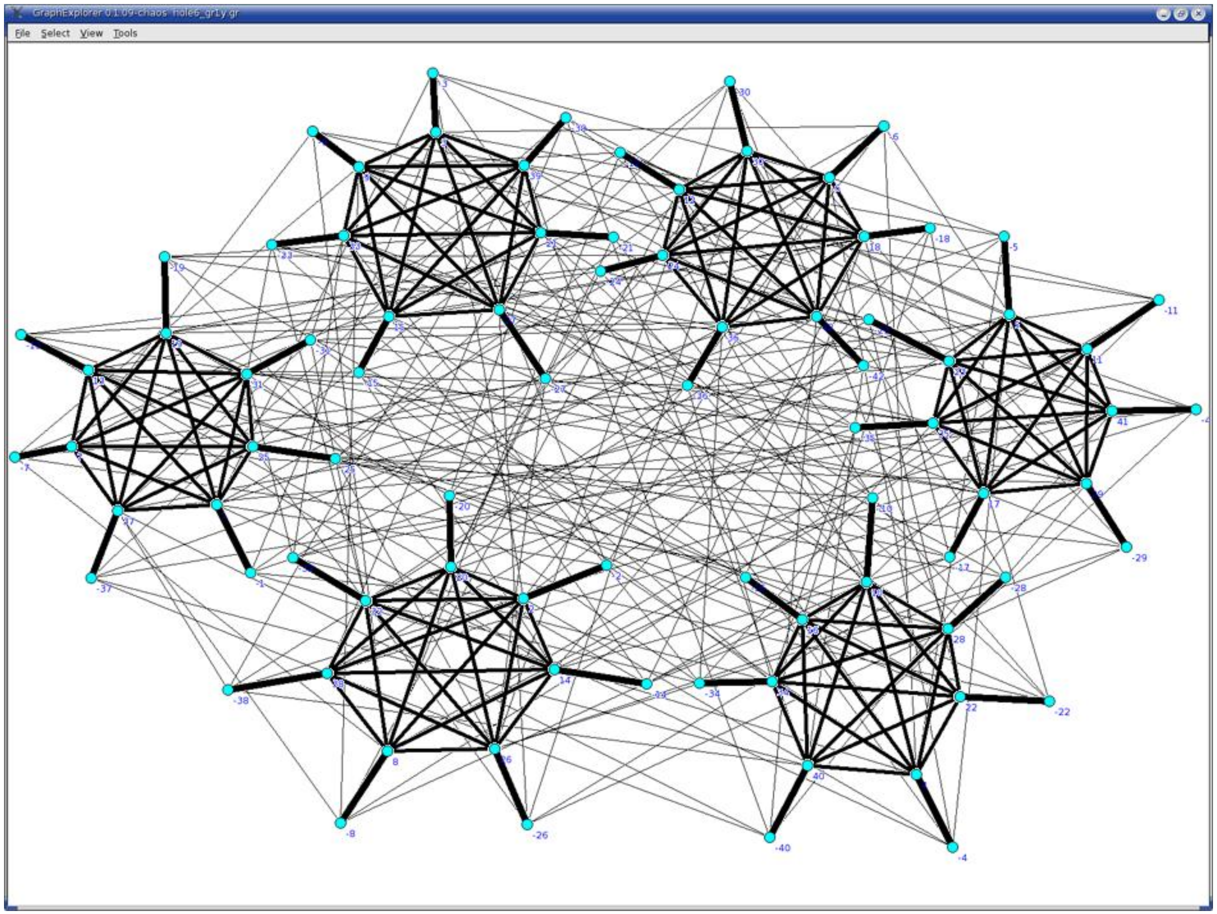
- After inferring **new conflicts – singleton UP**



Visualization (3)

using GraphExplorer software (Surynek, 2007-2010)

- After enforcing **clique consistency: UNSAT**



Complexity of Clique Consistency

- Construction of graph of conflicts
  - **polynomial** worst-case time
- Singleton unit propagation
  - **polynomial** worst-case time
  - however, may be too time consuming for large real-life problems
    - efficient propagation scheme base on 2-literal watching must be used
- Clique consistency with respect to a single sub-formula
  - **polynomial**
- **Problem:** clique consistency with respect to multiple sub-formulae
  - we cannot try all the sub-formulae
  - intelligent selection of promising sub-formulae must be done

Competitive Comparison

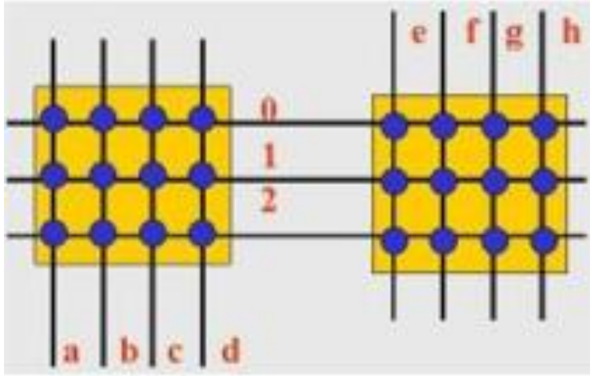
carried out in 2007

- **Tested** SAT solving **systems**
  - **MiniSAT**
  - **zChaff**
  - **HaifaSAT**

  winners in **SAT Competition 2005** and **SAT Race 2006**

  - selection criterion: **available source code**

- **Testing instances** (by Fadi Aloul)
  - **Pigeon Hole Principle**
  - **Urquhart** (resists resolution method)
  - **Field Programmable Gate Array**



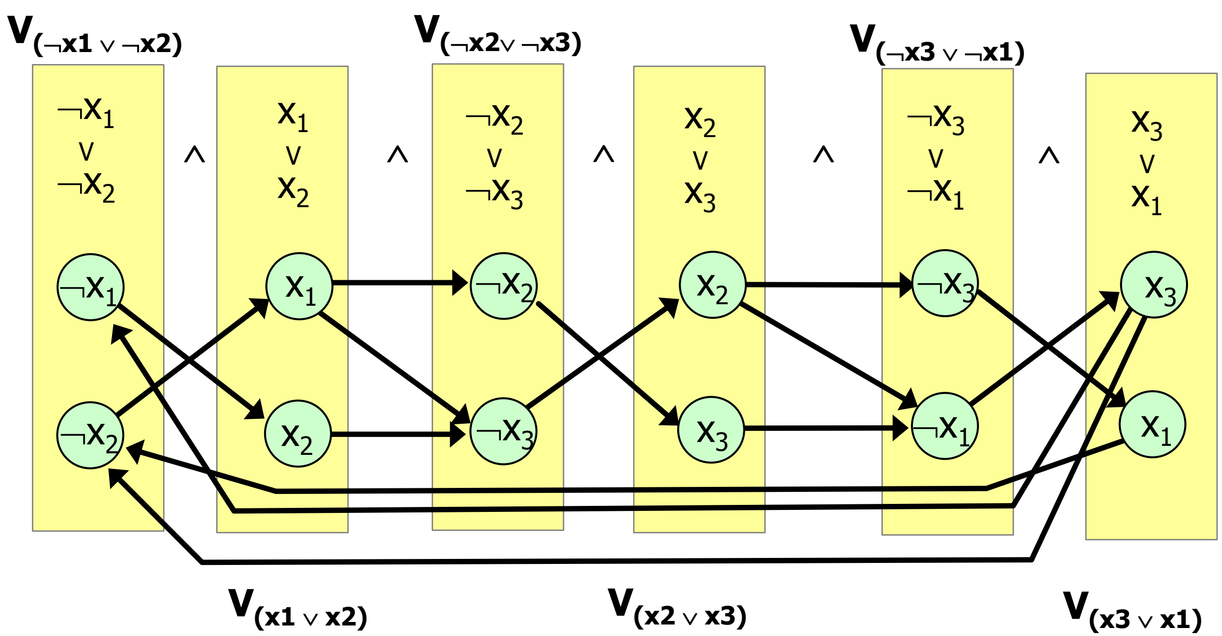
Experimental Evaluation

Instance	Decision (seconds)	Speedup ratio w.r.t. MiniSAT	Speedup ratio w.r.t zChaff	Speedup ratio w.r.t HaifaSAT
chnl10_11	**0.43**	**79.76**	**17.53**	*> 1395.34*
chnl10_12	**0.60**	**169.68**	**8.51**	*> 1000.00*
chnl10_13	**0.78**	**256.79**	**14.70**	*> 769.23*
chnl11_12	**0.70**	*> 857.14*	**47.84**	*> 857.14*
urq3_5	**130.15**	**0.73**	*N/A*	*N/A*
urq4_5	*> 600.00*	*N/A*	*N/A*	*N/A*
urq5_5	*> 600.00*	*N/A*	*N/A*	*N/A*
urq6_5	*> 600.00*	*N/A*	*N/A*	*N/A*
hole9	**0.08**	**45.5**	**18.25**	**5977.00**
hole10	**0.13**	**301.84**	**57.92**	*> 4615.38*
hole11	**0.20**	*> 3000.00*	**161.8**	*> 3000.00*
hole12	**0.30**	*> 2000.00*	**1240.6**	*> 2000.00*
fpga10_11	**0.46**	**97.32**	**27.34**	*> 1304.34*
fpga10_12	**0.64**	**186.34**	**52.84**	*> 937.50*
fpga10_13	**0.84**	**431.23**	**90.65**	*> 714.28*
fpga10_15	**1.39**	*> 431.65*	**197.72**	*> 431.65*

Opteron 1600 MHz, Mandriva Linux 10.1

Path-consistency in Literal Encoding (1)

- SAT as CSP: **Literal encoding** model (X,C,D)
  - X ... variables ↔ clauses, C ... constraints ↔ values standing for complementary literals are forbidden, D ... variable domains ↔ literals
- Interpret path-consistency in the CSP model of SAT as a **directed graph**
  - **vertices** ↔ values in domains, **edges** ↔ allowed pairs of values



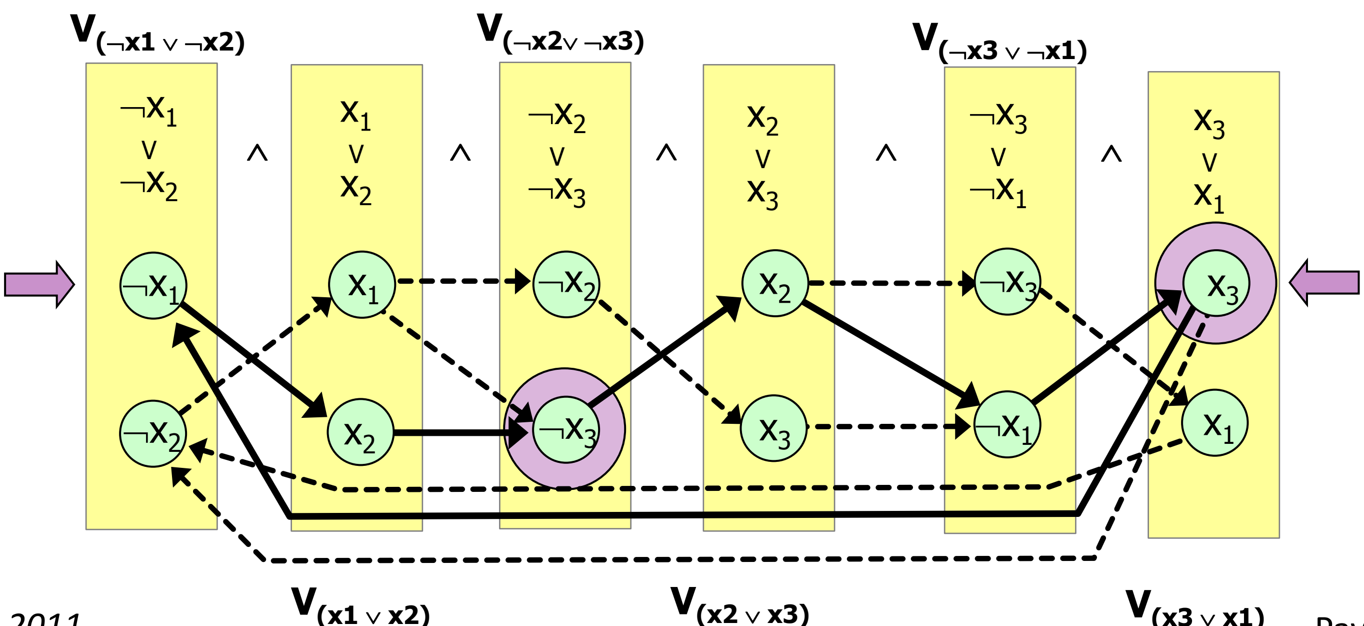
example:
$X=V_{(\neg x1 \vee \neg x2)}, V_{(x1 \vee x2)}, \ldots$

example:
$D(V_{(\neg x1 \vee \neg x2)})=\{\neg x_1, \neg x_2\}$

example:
$V_{(\neg x1 \vee \neg x2)} = \neg x_1$ and
$V_{(x1 \vee x2)} = x_1$
is forbidden

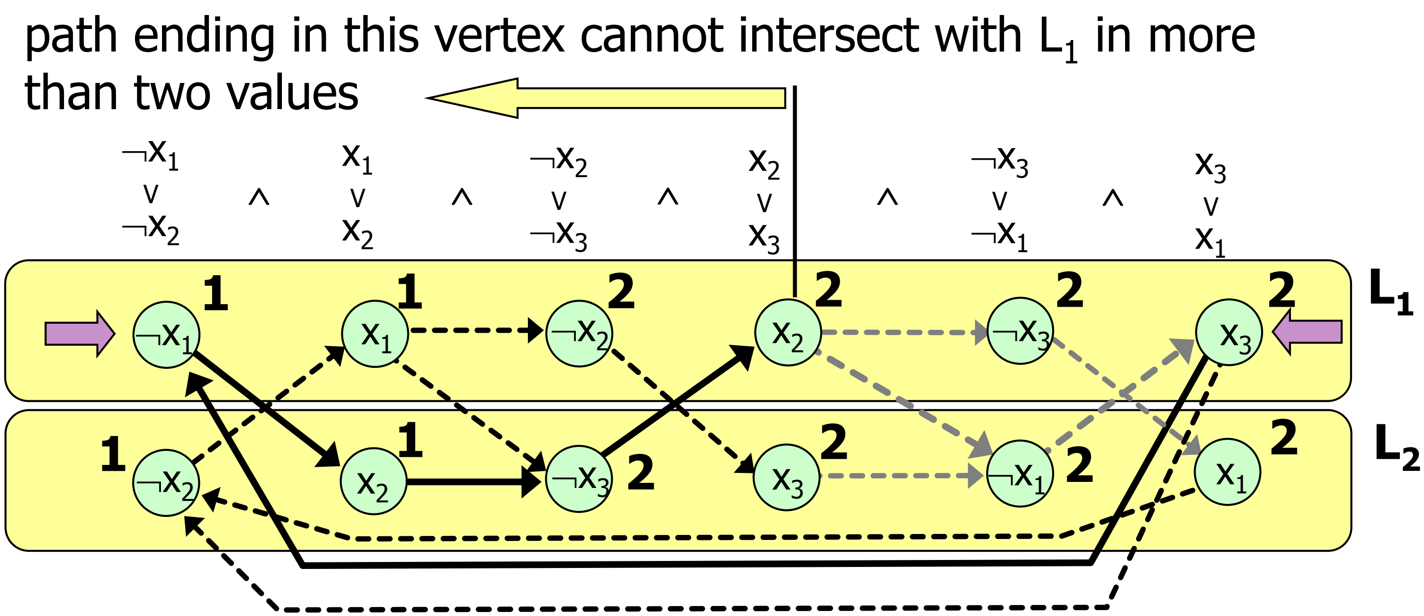
Path-consistency in Literal Encoding (2)

- Let us have a **sequence of variables (path)**
  - pair of values is **path-consistent** w.r.t. to the sequence if there is an oriented path connecting them in the graph interpretation going through the sequence and values itself are connected
- **Ignores** constraints between non-neighboring variables in the sequence of variables



Modified Path-Consistency for SAT

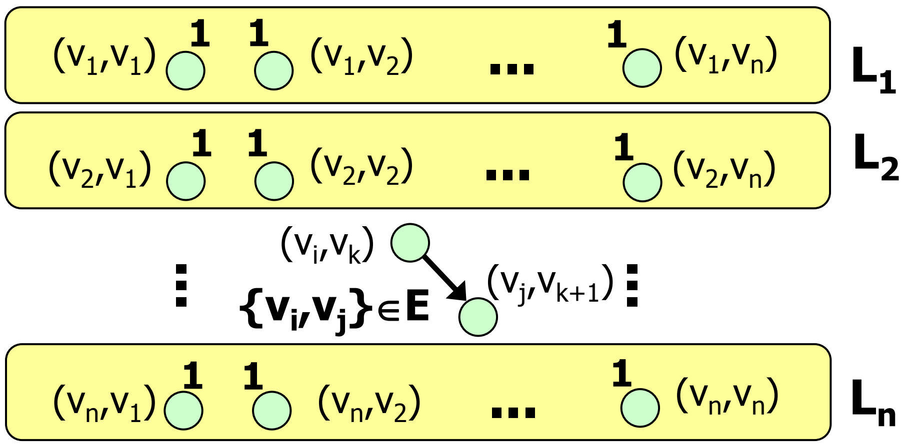
- Deduce **more information from constraints**
  - decompose values into **disjoint sets** (called **layers** ... $L_1, L_2, \ldots, L_M$)
  - **deduce more information** from constraints - calculate **maximum size of the intersection of the constructed path with individual layers –** denoted as χ
- Stronger restriction on paths ► **stronger propagation**



NP-completeness of the Modified Path Consistency

- Enforcing **modified path-consistency** is **difficult** (NP-complete)
  - The decision problem is whether there exists a path respecting the maximum sizes of intersections with individual layers.
- **Lemma:** The decision variant of the problem belongs to the NP class.
  - The path is of polynomial size with respect to the graph interpretation.
  - It can be checked in polynomial time whether the path conforms to maximum size of intersections with individual layers.
- **Lemma:** The existence of a **Hamiltonian path** in a graph is reducible to the existence of a path conforming to the maximum sizes of intersections with layers.

- **Main idea** of the proof: **G=(V,E)**, where $V=\{v_1,v_2,\ldots,v_n\}$
  (i) Construct an instance of modified path consistency in the form of a matrix
- **(ii)** Associate rows of the matrix with layers and set the maximum size of the intersection to 1



Pavel Surynek, 2011

Intersection Matrices

- An **intersection matrix** is defined for each value in the graph interpretation of path-consistency – it is denoted as ψ(v)
  - Let L_1, L_2, …, L_M be a layer decomposition of the graph interpretation
  - Let K be the number of variables involved in the path
  - ▶ The **intersection matrix** is of type M × K
- Intersection matrix **ψ(v)** w.r.t. a pair of values v_0 and v_K
  - $\boldsymbol{\psi(v)_{i,j}}$ represents the number of paths starting in v_0 and ending in v that **partially** conform to maximum sizes of intersection with layers such that they intersect with L_i j-times.
- It is not possible to enforce exact conformity to calculated maximum sizes of intersection with layers
  - Therefore we need to talk about partial conformity.

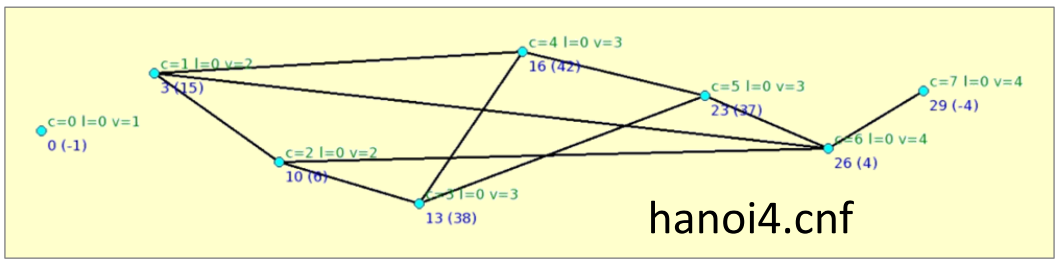
Intersection Matrices Update

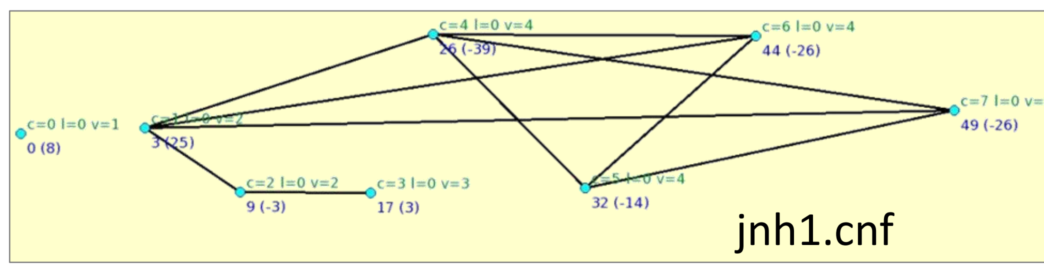
- **Intersection matrix** can be updated easily
  - $\psi(v)$ is calculated from $\psi(u_1)$, $\psi(u_2)$,..., $\psi(u_m)$ where u_1, u_2,..., u_m are a values from the domain of the **previous variable** in the path
- If it is detected that **no** of the paths starting in $\mathbf{v_0}$ and ending in **v** conforms to the maximum size of the intersection with the layer L_i such that $v \in L_i$ then $\psi(v)$ is set to 0 (matrix)
  - maximum intersection sizes with other layers cannot be violated since intersection size with them does no change
  - **relaxation:** paths that do not conform to maximum sizes of intersections with layers are propagated further

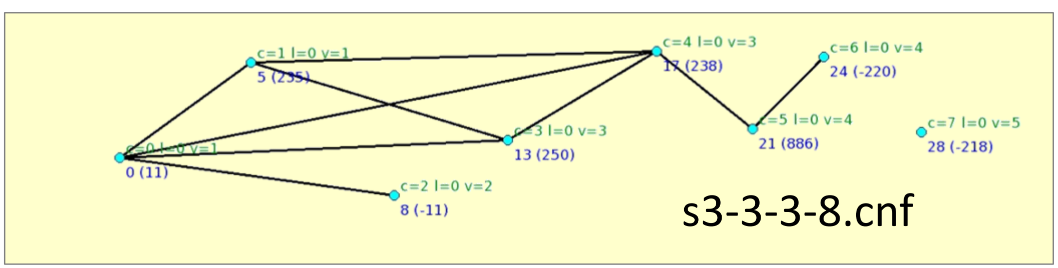
Visualization of Layers

using GraphExplorer software (Surynek, 2007-2010)

- Layer decomposition was constructed with several **most constrained clauses** (now: edges = **forbidden** pairs)
  - several benchmark problems from the **SAT Library**







Maximum Intersection Sizes

- Maximum intersection size is calculated using the maximum intersection size for the previous value in the layer
  - it is checked whether the intersection size can be increased by adding the current value

SAT instance	Maximum **intersection** with $L_1=[v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7]$							
	$X(v_0)$	$X(v_1)$	$X(v_2)$	$X(v_3)$	$X(v_4)$	$X(v_5)$	$X(v_6)$	$X(v_7)$
ais12.cnf	1	1	1	1	1	1	1	1
hanoi4.cnf	1	2	2	3	3	3	4	4
huge.cnf	1	1	2	2	2	2	3	3
jnh1.cnf	1	2	2	3	4	4	4	5
par16-1.cnf	1	1	1	2	2	2	2	2
par16-1-c.cnf	1	2	2	3	3	4	4	5
pret150_75.cnf	1	1	2	2	3	3	4	4
s3-3-3-8.cnf	1	1	2	3	3	4	4	5
ssa7552-160.cnf	1	1	2	3	4	4	5	6
sw100-5.cnf	1	1	2	2	2	2	3	3
Urq8_5.cnf	1	1	2	2	3	3	4	4
uuf250-0100.cnf	1	1	2	2	3	3	4	4

Experimental Evaluation (1)

SAT Problem	Number of variables	Number of clauses	Pairs filtered by standard PC	*Pairs filtered by modified PC*
bw_large.a	495	4675	22	*22*
hanoi4	718	4934	9	***10***
huge	459	7054	12	*12*
jnh2	100	850	135	***147***
logistics.a	828	6718	192	*192*
medium	116	953	177	***227***
par8-1-c	64	254	0	***19***
par8-2-c	68	270	0	***9***
par8-3-c	75	298	0	***100***
par16-1-c	317	1264	0	***11***
par16-2-c	349	1392	0	***7***
par16-3-c	334	1332	0	***7***
ssa0432/003	435	1027	81	***1598***
ssa2670/130	1359	3321	4	***2656***
ssa2670/141	986	2315	20	***8871***
ssa7552/038	1501	3575	16	***5652***
ssa7552/158	1363	3034	49	***2371***

- Comparison of the number of **filtered pairs of values**
  - several benchmark problems from the **SAT Library**
  - comparison of **PC** and **modified PC** enforced by the basic variant of intersection matrix update algorithm
  - **on some problems** modified PC is significantly **stronger**
  - runtime was slightly higher for modified PC

Experimental Evaluation (2)

Problem	#variables	#clauses	HaifaSat	Minisat2	Rsat_1_03	zChaff
bw_large.a	459	4675	1.0	1.0	1.0	1.0
hanoi4	718	4934	1.0	1.0	1.0	1.0
hanoi5	1931	14468	1.0	1.0	1.0	1.0
huge	459	7054	1.0	1.0	1.0	1.0
jnh2	100	850	1.0	1.0	1.0	**1.3**
logistics.a	828	6718	1.0	1.0	1.0	1.0
medium	116	953	1.0	1.0	0.8	0.9
par8-1-c	64	254	1.0	1.0	0.9	0.7
par8-2-c	68	270	0.9	**1.2**	0.7	0.8
par8-3-c	75	298	0.8	**1.4**	0.6	0.8
par16-1-c	317	1264	0.1	0.4	**2.2**	0.1
par16-2-c	349	1392	1.1	**2.3**	0.8	0.8
par16-3-c	334	1332	0.8	**1.4**	**6.6**	**1.6**
ssa0432-003	435	1027	1.0	**228.0**	**155.0**	**122.0**
ssa2670-130	1359	3321	**51.0**	**411.0**	**371.0**	**323.0**
ssa2670-141	986	2315	**289.0**	**429.0**	**455.0**	**489.0**
ssa7552-038	1501	3575	**190.0**	**226.0**	**173.0**	**238.0**
ssa7552-158	1363	3034	**114.0**	**129.0**	**151.0**	**312.0**

- **Improvement ratio** gained by preprocessing of SAT problems by modified PC in comparison with PC
  - the **number of decision steps** was measured
  - **some problems** were **successfully** preprocessed by modified PC

References

- Chmeiss, A., Jégou, P.: Efficient Constraint Propagation with Good Space Complexity. Proceedings of the Second International Conference on Principles and Practice of Constraint Programming (CP 1996), pp. 533-534, LNCS 1118, Springer, 1996.
- Cook, S. A.: The Complexity of Theorem Proving Procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971), pp. 151-158, ACM Press, 1971.
- Dowling, W., Gallier, J.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. Journal of Logic Programming, Volume 1 (3), 267-284, Elsevier Science Publishers, 1984.
- Holger, H. H., Stützle, T.: SATLIB: An Online Resource for Research on SAT. Proceedings of Theory and Applications of Satisfiability Testing, 4th International Conference (SAT 2000), pp.283-292, IOS Press, 2000, http://www.satlib.org, [October 2010].
- Mohr, R., Henderson, T. C.: Arc and Path Consistency Revisited. Artificial Intelligence, Volume 28 (2), 225-233, Elsevier Science Publishers, 1986.
- Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences, Volume 7, pp. 95-132, Elsevier, 1974.
- Petke, J., Jeavons, P.: Local Consistency and SAT-Solvers. CP 2010, pp. 398-413, Springer, 2010.
- Surynek, P.: Making Path Consistency Stronger for SAT. Proceedings of the Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming (CSCLP 2008), ISTC-CNR, 2008.