Application of Hierarchical Hybrid Encodings to Efficient Translation of CSPs to SAT

Van-Hau Nguyen

International Center for Computational Logic
TU Dresden

Joint with Miroslav N. Velev and Pedro Barahona

Outline

- Motivation
- Preliminaries
- The Log-Direct & Log-Order Encodings
- Comparisons & Experiments
- Conclusions & Future Works

Motivation – SAT solving

- Boolean Satisfiability (SAT) solving has been an increasing impact on applications in various areas
- SAT solving = SAT solvers + SAT encodings
  - SAT solvers have reached a high level of development and efficiency.
  - SAT encodings: … **limited** and **challenging**
- Many applications in computer science can be expressed as Constraint Satisfaction Problems (CSPs), while hardly any such problems are originally given by SAT formulas

➔ Translating CSPs into SAT !

Motivation – SAT encodings

- Direct encoding [deKleer89] (Support [Gent02]):
  - ☺ Propagation: Forward checking (Arc consistency), e.g. NP-SPEC [Cadoli05], CSP2SAT 4J[Berre08]
  - ☹ # variables: BIG
- Order encoding [Tamura06]
  - ☺ Propagation: bound, e.g. Sugar [Tamura06], BEE [Metodi12]
  - ☹ # variables: BIG
- Log encoding [Walsh00]
  - ☺ # variable: small
  - ☹ Propagation: less powerful, due to long clauses

➔ Studying the log-direct & the log-order encodings!

Preliminaries – CSP

- A Constraint Satisfaction Problem (CSP) is a triple $\{\mathcal{V}, \mathcal{D}, \mathcal{C}\}$, where:
  - $\mathcal{V}$ is a set of *k* multi-valued variables,
  - $\mathcal{D}$ is the set of their domains,
  - $\mathcal{C}$ is a set of m constraints.
- A *finite* CSP : a finite set of variables, each having a finite domain
- The CSP problem is to determine whether there exists one assignment that satisfies all the constraints
- We consider a CSP variable *v* with the domain *n* values

Preliminaries – SAT

- A Boolean Satisfiability problem (SAT)
  - A conjunction normal form (CNF) is a conjunction of ***clauses,*** defined on a set of Boolean ***variables***.
  - A clause is a disjunction of ***literals*** , where a literal is either a Boolean variable or its negation.
- A clause is
  - *satisfied* if at least one of its literals is assigned to *true*,
  - *unsatisfied* if all of its literals are assigned to *false*.
- The formula is *satisfiable* if there exists a truth assignment that satisfies all of its clauses, *unsatisfiable* otherwise
- The SAT problem is to determine whether a formula is *satisfiable*

Preliminaries – CSP2SAT

- The direct encoding [deKleer89] (support [Gent02])
  - Uses *n* propositional variables $d_v^i, 1 \le i \le n$ to encode a CSP *v* with a finite domain $\{v_1, v_2, \ldots, v_n\}$
  - Requires: at-least-one and at-most-one clauses
- The order encoding [Tamura06]
  - Uses a vector of *n* -1 propositional variables $o_v^1, \ldots, o_v^{n-1}$ to encode CSP *v* with a finite domain $\{v_1, v_2, \ldots, v_n\}$
  - Requires*:* the domain constraint (a non-increasing vector)

$$\bigwedge_{i=1}^{n-2} \neg(\neg o_v^i \wedge o_v^{i+1}) = \bigwedge_{i=1}^{n-2} (o_v^i \vee \neg o_v^{i+1})$$

  - The interval variables → the propagation of bounds [5, 23, 32]

Preliminaries – CSP2SAT

- The log encoding [Walsh00]
  - Uses $m = \lceil log_2 n \rceil$ Boolean variables.
  - Requires no at-least-one and at-most clauses, but prohibited-value clauses.

- Hierarchical Hybrid Encodings [Velev07]
  - A domain is recursively divided into smaller subdomains, until at the lowest level each subdomain contains a single a domain value. At each level of the hierarchy, one can choose a simple encoding and the number of subdomains on the next level.
  - The 12 simple encodings combined with a variety of structures led to a numerous way of translations of a domain to SAT → impractical

The Log-direct Encoding (1)

- Basic Idea

  The log-direct encoding is the first such hierarchical hybrid encoding, where the log encoding at level one has one indicator variable b_v dividing the domain into two subdomains represented at level two with the direct encoding by mean of Boolean "direct" variables.

- An assignment can be expressed

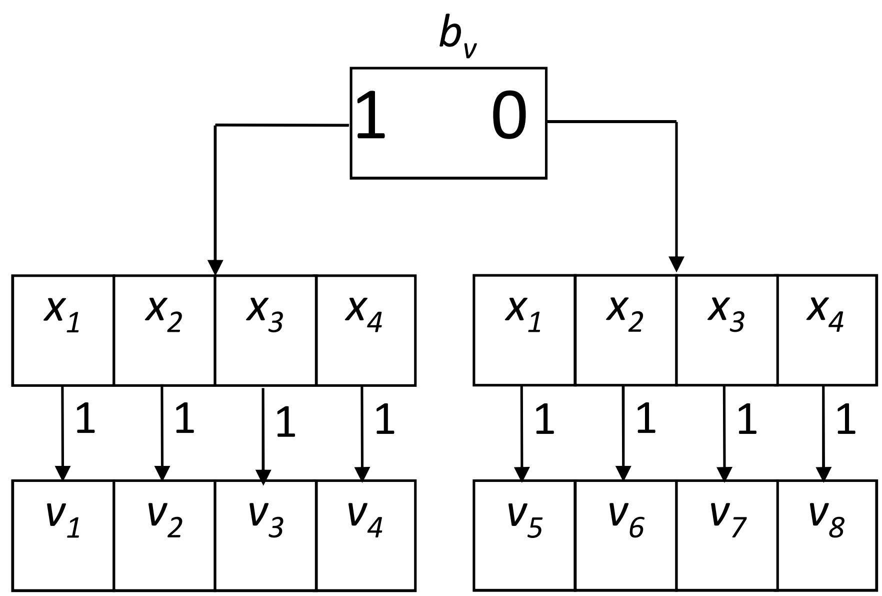
$$v = v_i \Leftrightarrow \begin{cases} b_v \wedge x_i & \text{if } 1 \leq i \leq \lfloor n/2 \rfloor \\ \neg b_v \wedge x_{i-\lfloor n/2 \rfloor} & \text{if } \lfloor n/2 \rfloor < i \leq n \end{cases}$$

The Log-direct Encoding (2)

- **Proposition 1**

  The log-direct encoding to index the domain values of some CSP variable when encoding a CSP onto an equivalent SAT problem is sound and complete.

- An illustration of encoding a CSP variable into SAT.



The Log-direct Encoding (3)

- Proposition 2

  The sparse encoding requires ***n*** *Boolean variables to encode a CSP variable v with n domains whereas the log-direct encoding requires only* ***n/2+1*** *Boolean variables.*

- Proposition 3

  Unit propagation applied to the log-direct encoding is incomparable to the direct encoding.

  Suppose CSP variables: *W, Y* with domain {1, …, 8}.

  - And a CSP constraint $W \neq 3 \vee Y \neq 5$:
    - The log-direct : $(\neg b_W \vee \neg x_3) \vee (b_Y \vee \neg x_1^Y)$
    - The direct : $\neg d_W^{\ 3} \vee \neg d_Y^{\ 5}$
    - Now Y = 5…
  - A CSP constraint $W \leq 5$
    - The log-direct : $b_W \vee x_1$
    - The direct: $d_W^{\ 1} \vee d_W^{\ 2} \vee d_W^{\ 3} \vee d_W^{\ 4} \vee d_W^{\ 5}$
    - Now $W \neq 5$…

The Log-direct Encoding (4)

- Proposition 4

  Unit propagation applied to the log-direct encoding is stronger than to the log encoding.

  1. If unit propagation commits to particular truth assignments on the log encoding, then unit propagation commits to the same truth assignments on the log-direct encoding.
  2. If unit propagation generates the empty clause in the log encoding then unit propagation generates the empty clause in the log-direct encoding then (but the reverse does not necessarily hold).

  - *Proof*: Similar to the proof of Theorem 15 [Walsh00]

The Log-order Encoding (1)

- Basic Idea

  The order-direct encoding is the first such hierarchical hybrid encoding, where the log encoding at level one has one indicator variable b_v dividing the domain into two subdomains represented at level two with an order encoding by mean of Boolean order variables.

- An assignment can be expressed

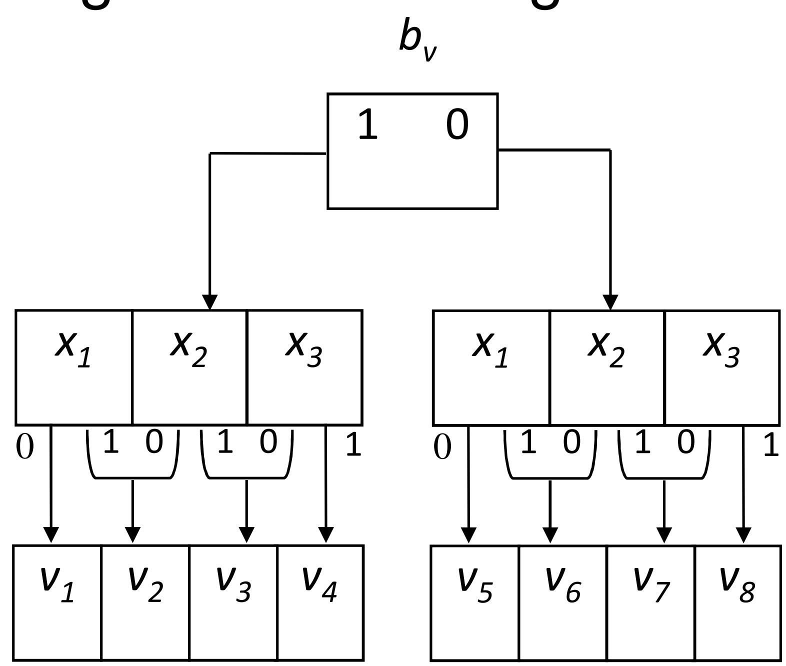
$$v = v_i \Leftrightarrow \begin{cases} b_v \wedge \neg x_1 & if\ i = 1 \\ b_v \wedge x_{i-1} \wedge \neg x_i & if\ 1 < i < \lfloor n/2 \rfloor \\ b_v \wedge x_{\lfloor n/2 \rfloor - 1} & if\ i = \lfloor n/2 \rfloor \\ \neg b_v \wedge \neg x_1 & if\ i = \lfloor n/2 \rfloor + 1 \\ \neg b_v \wedge x_{i - \lfloor n/2 \rfloor - 1} \wedge \neg x_{i - \lfloor n/2 \rfloor} & if\ \lfloor n/2 \rfloor + 1 < i < n \\ \neg b_v \wedge x_{\lceil n/2 \rceil - 1} & if\ i = n \end{cases}$$

The Log-order Encoding (2)

- Proposition 5

  The log-order encoding to index the domain values of some CSP variable when encoding a CSP onto an equivalent SAT problem is sound and complete.

- An illustration of the log-order encoding to encode a CSP variable into SAT.



The Log-order Encoding (3)

- Proposition 6

  *The sparse encoding requires **n-1** Boolean variables to encode a CSP variable v with n domains whereas the log-order encoding requires only **n/2** Boolean variables.*

- Proposition 7

  Unit propagation applied to the log-order encoding is incomparable to the order encoding.

  - *Proof: Similar to Proposition 3 by giving an appropriate example*

Comparisons

- Comparison of different encodings regarding to the number of variables and clauses required to encode a CSP variable with *n* value into SAT

Methods	# variables	# clauses
direct	n	$\sim n^2/2$
log-direct	$\sim n/2$	$\sim n^2/8$
order	$n\text{-}1$	$\sim n$
log-order	$\sim n/2$	$\sim n/2$

Experiments – Pigeon Hole Problems

- Proving that ***n*** pigeons can not fit in ***n-1*** holes.

Instance	*direct*	*l.direct*	*Speedup*	*order*	*l.order*	*Speedup*
11	18.2	**0.8**	22.8×	3.3	**0.9**	**3.7×**
12	170.9	**5.3**	32.2×	15.2	**4.2**	3.8×
13	2,033.1	**22.9**	88.8×	72.0	**15.5**	4.6×
14	28,332.9	**125.2**	226.3×	1,013.4	**65.8**	15.4×
15	>86,400.0	**661.2**	>130.7×	7,394.6	**232.2**	31.8×
Min.	18.2	0.8	22.8×	3.3	0.9	3.7×
Max.	>86,400.0	661.2	226.3×	7,394.6	232.2	31.8×
Average	>7,638.8	163.1	>100.1×	1,699.7	63.7	11.9×

Experiments – Graph Coloring Problems

- Requiring an assignment of colors to the vertices of an undirected graph, such that no two adjacent vertices share the same color.

- The instances taken from [Color03]

Instance	*K*	*direct*	*l.direct*	*Speedup*	*order*	*l.order*	*Speedup*
DSJC125.9	9	35.6	**0.6**	57.5×	0.7	**0.5**	1.4×
	10	274.8	**1.6**	171.8×	2.3	**1.3**	1.8×
	11	10,839.2	**7.0**	1548.5×	8.1	**3.8**	2.1×
DSJC250.9	9	71.9	**5.4**	13.1×	3.6	**3.4**	1.1×
	10	376.8	**9.2**	41.0×	7.4	**4.5**	1.6×
	11	11,150.8	**213.8**	52.2×	**42.2**	142.0	0.3×
miles750	9	15.6	**0.3**	58.0×	0.6	**0.2**	2.8×
	10	187.5	**0.7**	267.9×	1.9	**0.6**	3.2×
	11	3,214.1	**4.5**	714.2×	15.5	**3.1**	5.0×
miles1000	9	18.8	**0.3**	49.6×	0.4	**0.3**	1.1×
	10	319.0	**0.9**	354.4×	3.0	**0.9**	3.3×
	11	8,932.9	**5.1**	1751.5	21.0	**3.9**	5.4×
miles1500	9	26.8	**0.6**	44.8×	0.9	**0.5**	1.7×
	10	463.4	**1.3**	356.5×	2.5	**0.9**	2.8×
	11	11,295.2	**6.8**	1661.1×	20.7	**3.6**	5.8×
queen12_12	9	6.8	**0.3**	21.5×	0.5	**0.2**	2.5×
	10	28.6	**0.7**	38.7×	1.7	**0.7**	2.2×
	11	152.3	**4.7**	32.3×	5.2	**3.9**	1.3×
queen13_13	9	8.4	**0.3**	23.4×	0.7	**0.3**	2.4×
	10	63.7	**0.8**	75.0×	1.9	**0.9**	2.1×
	11	473.7	**4.7**	99.7×	8.3	**3.5**	2.4×
queen14_14	9	11.75	**0.4**	28.7×	0.7	**0.3**	1.9×
	10	141.4	**0.9**	157.1×	2.2	**0.9**	2.4×
	11	967.3	**4.8**	201.0×	15.3	**0.9**	17.0×
queen15_15	9	19.6	**0.4**	49.0×	0.9	**0.4**	2.3×
	10	214.9	**1.1**	195.4×	3.2	**1.0**	3.2×
	11	1237.3	**5.3**	230.0×	15.3	**4.0**	3.8×
queen16_16	9	16.2	**0.6**	26.6×	0.6	**0.5**	1.2×
	10	128.9	**1.2**	107.4×	3.7	**1.1**	3.4×
	11	2,194.9	**5.1**	430.4×	60.4	**5.1**	11.8×
school1	9	44.6	**1.6**	27.6×	1.3	**1.2**	1.1×
	10	345.7	**4.5**	76.8×	5.2	**2.0**	2.6×
	11	3,858.0	**10.2**	378.2×	9.8	**7.6**	1.3×
2-FullIns 5	5	**0.3**	1.7	0.2×	2.0	**0.2**	8.8×
	6	**42.5**	3,762.6	0.01×	4,842.5	**16.0**	301.5×
5-FullIns	7	**0.7**	6.9	0.1×	2.0	**0.4**	5.1×
	8	**109.65**	3,966.7	0.01×	4,235.8	**12.4**	341.6×
Min.		0.3	0.3	0.01×	0.4	0.2	0.30×
Max.		11,295.2	3,966.7	1,751.5×	4,842.5	142.0	341.6×
Average		1,555.7	217.4	252.5×	252.7	6.3	20.6×

Experiments – *Open Shop Problems*

- Finding the minimize the makespan such that given *n* jobs and *m* workstations, each job has to be processed on a workstation at least once

- The instances taken from [Taillard]

Instance	*M*	*S/U*	*direct*	*l.direct*	*Speedup*	*order*	*l.order*	*Speedup*
4_1	192	U	19.4	**1.3**	14.9×	**0.09**	0.15	0.6×
	193	S	20.1	**1.3**	15.5×	**0.17**	0.23	0.7×
4_2	235	U	35.7	**1.9**	18.8×	**0.11**	0.22	0.5×
	236	S	37.3	**1.7**	21.9×	**0.14**	0.20	0.7×
4_3	270	U	65.9	**2.7**	24.4×	**0.25**	0.31	0.8×
	271	S	70.1	**2.6**	27.0×	**0.16**	0.24	0.7×
4_4	249	U	42.8	**2.3**	18.6×	**0.17**	0.29	0.6×
	250	S	44.1	**1.7**	25.9×	**0.16**	0.25	0.6×
4_5	294	U	77.1	**3.3**	23.4×	**0.19**	0.35	0.5×
	295	S	78.5	**2.7**	29.1×	**0.12**	0.20	0.6×
5_1	299	U	127.5	**6.2**	20.6 ×	0.71	**0.65**	1.1×
	300	S	126.7	**7.1**	17.8 ×	0.72	**0.69**	1.0×
5_2	261	U	130.9	**8.8**	14.9×	**0.84**	0.99	0.8×
	262	S	129.5	**7.7**	16.8×	0.75	**0.59**	1.3×
5_3	327	S	190.1	**16.9**	11.2×	**0.57**	1.1	0.5×
	328	S	198.4	**14.7**	13.5×	2.0	**1.8**	1.1×
5_4	309	U	257.8	**17.4**	14.8×	**2.3**	2.6	0.9×
	310	S	176.6	**13.5**	13.1×	1.2	1.7	0.7×
5_5	325	U	408.6	**20.8**	19.6×	4.4	**4.1**	1.1×
	326	S	174.2	**19.0**	9.2×	**1.3**	2.7	0.5×
7_1	436	S	252.2	**130.0**	1.9×	**7.1**	8.1	0.9×
	437	S	296.4	**161.0**	1.8×	11.7	**9.0**	1.1×
7_2	444	S	213.4	**168.9**	1.3×	12.8	**12.1**	1.1×
	445	S	265.6	**144.2**	1.8×	**5.1**	10.7	0.5×
7_3	470	S	669.1	**349.3**	1.9×	53.8	**28.7**	1.9×
	471	S	750.6	**482.9**	1.6×	51.0	**40.8**	1.3×
7_4	463	S	637.1	**271.5**	2.3×	19.8	**12.4**	1.6×
	464	S	275.5	**160.7**	1.7×	10.6	**5.7**	1.9×
7_5	416	S	268.3	**142.0**	1.9×	27.3	**15.1**	1.8×
	417	S	320.4	**99.7**	3.2×	11.9	**4.9**	2.4×
Min.			19.4	1.3	1.3×	0.07	0.15	0.5×
Max.			750.6	482.9	29.1×	53.8	40.8	2.4×
Average			186.9	75.5	13.0×	6.1	5.1	1.0×

Experiments – All Interval Serial Problems

- Arranging a permutation of the ***n*** integers ranging from differences between adjacent numbers are also a permutation, of the numbers from ***1*** to ***n-1*** [csplib].

- Finding all solutions

Instance	*direct*	*l.direct*	*Speedup*	*order*	*l.order*	*Speedup*	*Sol*
13	252.0	**26.2**	9.6×	**6.0**	18.3	0.3×	3200
14	1,787.5	**162.5**	11.0×	**16.0**	58.2	0.3×	9912
15	13,660.0	**206.0**	66.3×	**52.7**	265.8	0.2×	25592
16	>86,400.0	**742.4**	>116.4×	**107.4**	702.5	0.2×	55920
Min.	252.0	26.2	9.6×	6.0	18.3	0.2×	
Max.	>86,400.0	742.4	>116.4×	107.4	702.5	0.3×	
Average	>25,524.9	284.3	>50.8×	45.5	261.2	0.3×	

Conclusions

- Although we used only clasp solver, but similar results were obtained for other solvers (lingeling, riss3G).

- The *log-direct encoding* (mentioned by [Velev07] but not tested) significantly outperforms the *direct encoding*, with runtime speedups of one to two order of magnitude (increasing with the size of the instances).

- The *log-order encoding*, a hierarchical hybrid encoding based on a new combination of simple encodings. In general, the log-order encoding is comparable with the *order encoding.*

➔ Two simple encoding with the powerful propagation

Future Works

- Further study the use of the log-direct and log-order encodings for a wide variety of real-life problems

- Features of these problems that might be more efficiently explored by the different encodings, namely with large domains, where more than one indicator variable at the first level of encoding might be more suitable

- We will also develop a SAT-based CSP solver, like Sugar [Tamura09] with advanced improvements [Metodi12]

Thank you for your attention!

References

[Color03] A Computational Symposium at Cornell University, Ithaca, NY, USA, 2002. http://mat.gsia.cmu.edu/COLOR03/

[Berre08] Berre, D.L., Lynce, I.: CSP2SAT4J: A simple CSP to SAT translator. In: Proceedings of the Second International CSP Solver Competition (2008)

[Cadoli05] Cadoli, M., Schaerf, A.: Compiling problem specifications into SAT. Artif. Intell. 162(1-2), 89–120 (2005)

[csplib] Hnich, B., Miguel, I., Gent, I.P., Walsh, T.: CSPlib is a library of test problems for constraint solvers. http://www.csplib.org/

[Gent02] Gent, I.: Arc consistency in SAT. In fifteenth European Conference on Artificial Intelligence pp. 121–125 (2002)

[deKleer89] J. de Kleer, "A Comparison of ATMS and CSP Techniques," 11th IJCAI. International Joint Conference on Artificial Intelligence, August 1989.

[Metodi12] Metodi, A., Codish, M.: Compiling finite domain constraints to SAT with BEE. Theory and Practice of Logic Programming 12(4-5), 465–483 (2012)

References

[Tamura06] Tamura, N., Taga, A., Kitagawa, S., & Banbara, M. Compiling finite linear CSP into SAT. In *Proceedings of the 12th international conference on principles and practice of constraint programming (CP 2006) (pp. 590–603).*

[Velev07] Velev, M.N.: Exploiting hierarchy and structure to efficiently solve graph coloring as SAT. In: International Conference on Computer-Aided Design (ICCAD'07), November 5-8, 2007, San Jose, CA, USA. pp. 135–142. IEEE (2007)

[Velev09a] Velev, M.N., Gao, P.: Efficient SAT techniques for absolute encoding of permutation problems: Application to hamiltonian cycles. In: Eighth Symposium on Abstraction, Reformultion, and Approximation, SARA 2009, Lake Arrowhead, California, USA, 8-10 August 2009. AAAI (2009)

[Velev09b] Velev, M.N., Gao, P.: Exploiting hierarchical encodings of equality to design independent strategies in parallel SMT decision procedures for a logic of equality. In: IEEE International High Level Design Validation and Test Workshop, HLDVT 2009, San Francisco, CA, USA, 4-6 November 2009. pp. 8–13 (2009)

[Walsh00] Walsh, T.: SAT v CSP. In: Principles and Practice of Constraint Programming - CP2000. LNCS, vol. 1894, pp. 441–456 (2000)