Mapping Problems with Finite-Domain Variables into Problems with Boolean Variables[⋆]

Carlos Ansótegui and Felip Manyà

Computer Science Department
Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
{carlos,felip}@eup.udl.es

Abstract. We define a collection of mappings that transform many-valued clausal forms into satisfiability equivalent Boolean clausal forms, analyze their complexity and evaluate them empirically on a set of benchmarks with state-of-the-art SAT solvers. Our results provide empirical evidence that encoding combinatorial problems with the mappings defined here can lead to substantial performance improvements in complete SAT solvers.

1 Introduction

In the last few years, the AI community has investigated the generic problem solving approach which consists of modeling hard combinatorial problems as instances of the propositional satisfiability problem (SAT) and then solving the resulting encodings with algorithms for SAT. The success in solving SAT-encoded problems depends on both the SAT solver and the SAT encoding used. While there has been a tremendous advance in the design and implementation of SAT solvers, our understanding of SAT encodings is very limited and is yet a challenge for the AI community working on propositional reasoning.

In this paper we define a collection of mappings that transform many-valued clausal forms into satisfiability equivalent Boolean clausal forms and analyze their complexity. Given a combinatorial problem encoded as a many-valued clausal form, the mappings defined allow us to generate six different Boolean SAT encodings. We evaluated empirically the Boolean SAT encodings generated for a number of combinatorial problems (graph coloring, random binary CSPs, pigeon hole, and all interval series) using Chaff [21] and Siege_v4.[1] Our results provide empirical evidence that encoding combinatorial problems with the mappings defined here can lead to substantial performance improvements in complete SAT solvers. The behaviour of different SAT encodings of graph coloring and all interval series instances on local search solvers was analyzed in [1, 23].

These results are part of a research program about many-valued satisfiability that our research group has developed during the last decade (see e.g. [2, 5, 9, 11, 18, 20]). Our research program is aimed at bridging the gap between Boolean SAT encodings and constraint satisfaction formalisms. The challenge is to combine the inherent efficiencies of Boolean SAT solvers operating on uniform encodings with the much more compact and natural representations, and more sophisticated propagation techniques of CSP formalisms.

We have used before mappings between many-valued clausal forms and Boolean clausal forms to identify new polynomially solvable many-valued SAT problems [7, 19], to known which additional deductive machinery is required to design many-valued SAT solvers from Boolean SAT solvers [7, 10], and to solve many-valued SAT encodings with Boolean SAT solvers [3, 4]. We invite the reader to consult two survey papers [8, 17] that contain a summary of our previous work.

The paper is structured as follows. In Section 2, we formally define the syntax and semantics of the many-valued clausal forms used in the paper. In Section 3, we define six mappings that transform many-valued clausal forms into satisfiability equivalent Boolean clausal forms. In Section 4, we report the empirical investigation conducted to assess the performance of those mappings.

⋆ Research partially supported by projects TIC2001-1577-C03-03 and TIC2003-00950 funded by the *Ministerio de Ciencia y Tecnología*. We thank Carla Gomes for allowing us to use computational resources of the Intelligent Information Systems Institute (Cornell University).

1 The SAT solver Siege_v4 is publicly available at http://www.cs.sfu.ca/~ loryan/personal

2 Many-Valued Formulas

We first formally define the syntax and semantics of signed CNF formulas, and then present monosigned and regular CNF formulas, which are the subclasses of signed CNF formulas that are considered in this paper.

Definition 1. *A* truth value set *N is a non-empty finite set $\{i_1, i_2, \ldots, i_n\}$ where $n \in \mathbb{N}$. The cardinality of N is denoted by $|N|$. A total order $\leq$ is associated with N, which may be the empty order.*

Definition 2. *A* sign *is a set $S \subseteq N$ of truth values. A* signed literal *is an expression of the form $S:p$ where S is a sign and p is a propositional variable. The* complement *of a signed literal $S:p$, denoted by $\overline{S}:p$, is $(N \setminus S):p$. A* signed clause *is a disjunction of signed literals. A* signed CNF formula *is a conjunction of signed clauses. The* size *of a signed clause C, denoted by $|C|$, is the total number of literals occurring in C, and the size of a signed CNF formula Γ, denoted by $|\Gamma|$, is the sum of the sizes of the clauses of Γ.*

Definition 3. *An* interpretation *is a mapping that assigns to every propositional variable an element of the truth value set. An interpretation I* satisfies *a signed literal $S:p$ iff $I(p) \in S$,* satisfies *a signed clause C iff it satisfies at least one of the signed literals in C, and* satisfies *a signed CNF formula Γ iff it satisfies all clauses in Γ. A signed CNF formula is* satisfiable *iff it is satisfied by at least one interpretation; otherwise it is* unsatisfiable.

Definition 4. *A sign S is* monosigned *if it either is a singleton (i.e. it contains exactly one truth value) or the complement of a singleton. A monosigned sign S is* positive *if it is identical to $\{i\}:p$, and is* negative *if it is identical to $\{\bar{i}\}:p$ for some $i \in N$. A signed literal $S:p$ is a* monosigned literal *if its sign S is monosigned. A signed clause (a signed CNF formula) is a* monosigned clause *(a* monosigned CNF formula*) if all its literals are monosigned.*

Definition 5. *Given a monosigned CNF formula Γ, the* domain of a variable *p occurring in Γ is $N_\Gamma(p) = \{i \in N \,|\, \{i\}:p \text{ or } \{\bar{i}\}:p \text{ occur in } \Gamma\}$ if $N_\Gamma(p) = N$, and $N_\Gamma(p) \cup \{j\}$, where j is any element of $N \setminus N_\Gamma(p)$, otherwise. The* Boolean signature *of Γ is $\Sigma = \{\{i\}:p \,|\, \{i\}:p \text{ or } \{\bar{i}\}:p \text{ occur in } \Gamma\}$.*

Definition 6. *For all $i \in N$, let $\uparrow i$ denote the sign $\{j \in N \mid j \geq i\}$, where $\leq$ is the total order associated with N, and let $\overline{\uparrow i}$ denote the complement of $\uparrow i$. A sign S is* regular *if it either is identical to $\uparrow i$ (positive) or to $\overline{\uparrow i}$ (negative) for some $i \in N$. A signed literal $S:p$ is a* regular literal *if its sign S is regular. A signed clause (a signed CNF formula) is a* regular clause *(a* regular CNF formula*) if all its literals are regular.*

Definition 7. *Given a regular CNF formula Γ, the* domain of a variable *p occurring in Γ is $N_\Gamma(p) = \{i \in N \,|\, \uparrow i:p \text{ or } \overline{\uparrow i}:p \text{ occur in } \Gamma\}$. The* Boolean signature *of Γ is $\Sigma = \{\uparrow i:p \,|\, \uparrow i: p \text{ or } \overline{\uparrow i}:p \text{ occur in } \Gamma\}$.*

Example 1. Suppose that $N = \{1,2,3,4\}$. Then, we have that the signed clause $\{1,2,3\}:p_1 \vee \{4\}:p_2$ can be represented as a monosigned clause by $\{\overline{4}\}:p_1 \vee \{4\}:p_2$, and as a regular clause by $\overline{\uparrow 4}:p_1 \vee \uparrow 4:p_2$.

Signed CNF formulas and their subclasses have been studied since the early 90's by the research community working on automated theorem proving in many-valued logics [6, 13, 15, 16, 22]. A few years later, Frisch and Peugniez [14] used the term non-Boolean formulas to refer to signed CNF formulas.

3 Mappings

We define a number of mappings that transform a monosigned CNF formula into a satisfiability equivalent Boolean CNF formula. In the most straightforward mappings, the derived formula consists of the input monosigned CNF formula under Boolean semantics (i.e., monosigned literals are interpreted as Boolean literals, and the notion of satisfiability is Boolean) plus a set of clauses that link many-valued interpretations with Boolean interpretations. The additional clauses ensure that exactly one of the literals of the Boolean signature of the monosigned CNF formula which correspond to a certain many-valued variable evaluates to true under Boolean semantics. We consider several cases: using only the Boolean signature of the monosigned CNF formula; extending the Boolean signature with regular literals under Boolean semantics; and extending the Boolean signature with a logarithmic number of Boolean variables for each many-valued variable (i.e., using a logarithmic encoding of the many-valued variables). In the most involved mappings, monosigned literals are replaced by their regular or logarithmic encoding in the input monosigned CNF formula, and its Boolean signature is replaced by a regular or logarithmic signature.

We analyze the complexity of the Boolean CNF formula derived by each mapping as a function of the size of the input monosigned CNF formula and the cardinality of the truth value set.

3.1 Standard mapping (S)

The most straightforward mapping consists of dealing with the Boolean signature of the input monosigned CNF formula. In the standard (S) mapping, each positive monosigned literal of the input monosigned CNF formula is taken as a Boolean variable, and each negative monosigned literal is replaced with the negation of its complement and is taken as a negative Boolean literal; i.e., we take the input monosigned CNF formula under Boolean semantics. Moreover, we add for each many-valued variable p, a clause that states that p takes at least one value of its domain (ALO clause) and a set of clauses that state that p takes at most one value of its domain (AMO clauses). Assume that the domain of p in the input monosigned CNF formula Γ is $N_\Gamma(p) = \{i_1, \ldots, i_{m(p)}\}$. Then, the ALO clause is $\{i_1\}:p \vee \cdots \vee \{i_{m(p)}\}:p$, and the set of AMO clauses contains a clause $\neg(\{i_j\}:p) \vee \neg(\{i_k\}:p)$ for all j and k such that $1 \leq i < j \leq m(p)$.

The size of the SAT instance generated by mapping S from a monosigned CNF formula Γ is in $\mathcal{O}(|\Gamma|\,|N|^2)$: The size of the instance generated by S is the sum of the size of Γ, denoted by $|\Gamma|$, plus the sum of the size of the ALO clauses and the size of the AMO clauses. For every many-valued variable p, there is an ALO clauses of size $|N_\Gamma(p)|$, where $|N_\Gamma(p)|$ is the size of the domain of p. If the number of distinct many-valued variables occurring in Γ is var, the size of all the ALO clauses is in $\mathcal{O}(var\,|N|)$. For every many-valued variable p, there are $\frac{|N_\Gamma(p)|(|N_\Gamma(p)|-1)}{2}$ AMO clauses of size two, and the size of all the AMO clauses is in $\mathcal{O}(var\,|N|^2)$. Therefore, the size of the instance generated by S is in $\mathcal{O}(|\Gamma| + var\,|N|^2)$. Since $|\Gamma| \geq var$, the size of the instance generated by S is in $\mathcal{O}(|\Gamma|\,|N|^2)$.

3.2 Full logarithmic mapping (FL)

In the full logarithmic (FL) mapping, a logarithmic encoding is used to represent a many-valued variable as a Boolean variable. To encode a many-valued variable p, using a base 2 encoding, only $\lceil \log_2 |N_\Gamma(p)| \rceil$ Boolean variables are required. For example, if p has domain $\{1, 2, 3, 4\}$, then the monosigned literal $\{1\} : p$ is mapped to $\neg p^2 \wedge \neg p^1$, the monosigned literal $\{2\} : p$ is mapped to $\neg p^2 \wedge p^1$, the monosigned literal $\{3\}:p$ is mapped to $p^2 \wedge \neg p^1$, and the monosigned literal $\{4\}:p$ is mapped to $p^2 \wedge p^1$. If the size of the domain of p is not a power of 2, then two combinations are mapped to some monosigned literals. For example, if the domain of p is $\{1, 2, 3\}$, then $\{1\}:p$ is mapped to $\neg p^2$ (which subsumes $\neg p^2 \wedge p^1$ and $\neg p^2 \wedge \neg p^1$), $\{2\}:p$ is mapped to $p^2 \wedge \neg p^1$, and $\{3\}:p$ is mapped to $p^2 \wedge p^1$.

Given a monosigned CNF formula Γ, the signature of mapping FL is $\Sigma = \{p^j \mid 1 \leq j \leq \lceil \log_2 |N_\Gamma(p)| \rceil, \, p \text{ occurs in } \Sigma\}$, each positive monosigned literal occurring in the input monosigned CNF formula is replaced with its logarithmic encoding, and each negative monosigned literal of the form $\{\bar{i}\}:p$ is replaced with the negation of the logarithmic encoding of $\{i\}:p$.

The size of the SAT instance generated by mapping FL is, in the worst case, exponentially larger than the size of the input monosigned CNF formula. The problem is that we must apply distributivity to get a clausal form when we encode positive monosigned literals. To overcome that drawback, Frisch and Peugniez [14] defined the logarithmic mapping.

3.3 Logarithmic mapping (L)

Frisch and Peugniez [14] defined the logarithmic (L) mapping, which combines mapping S and mapping FL. Given a monosigned formula Γ, the signature of mapping L is the union of the Boolean signature and the signature of mapping FL. The Boolean CNF formula derived by mapping L is formed by Γ plus an additional set of clauses that link monosigned literals with the logarithmic encoding; this way they avoid incorporating the ALO and AMO clauses. For example, if the many-valued variable p has domain $\{1, 2, 3, 4\}$, then they add the following clauses to link the monosigned literals containing variable p with their logarithmic encoding:

$$\{1\}\!:\!p \leftrightarrow \neg p^2 \wedge \neg p^1,\ \{2\}\!:\!p \leftrightarrow \neg p^2 \wedge p^1,\ \{3\}\!:\!p \leftrightarrow p^2 \wedge \neg p^1,\ \{4\}\!:\!p \leftrightarrow p^2 \wedge \neg p^1$$

Note that, with the ALO and AMO clauses, the number of clauses needed in mapping S to state that a many-valued variable takes exactly one value from its domain is in $\mathcal{O}(|N|^2)$, but with the previous transformation the number of clauses needed is in $\mathcal{O}(|N| \log_2 |N|)$. The size of the SAT instance generated by mapping L from a monosigned CNF formula Γ is in $\mathcal{O}(|\Gamma| \log_2 |N|)$.

3.4 Full regular mapping (FR)

Béjar, Hähnle and Manyà [10] defined the full regular (FR) mapping, which transforms a regular CNF formula Γ into a satisfiability equivalent Boolean CNF formula whose size is in $\mathcal{O}(|\Gamma|)$. In this section we reformulate mapping FR in the case that the input formula is a monosigned CNF formula instead of a regular CNF formula.

Given a regular CNF formula Γ, the signature of mapping FR is $\Sigma = \{\uparrow i : p \mid \uparrow i : p \text{ or } \overline{\uparrow i} : p \text{ occur in } \Gamma\}$; i.e., the Boolean signature of Γ. In mapping FR, each positive regular literal is taken as a positive Boolean literal, and each negative regular literal is taken as a negative Boolean literal. Moreover, we add, for each many-valued variable p, a set of clauses that link regular interpretations with Boolean interpretations [10]. Assume that the domain of p in the input regular CNF formula Γ is $N_\Gamma(p) = \{i_1, \ldots, i_{m(p)}\}$ and $i_1 \le i_2 \le \cdots \le i_{m(p)}$ under the order $\le$ associated with N. Then, the set of clauses added is:

$$\{\neg(\uparrow i_{(j+1)} : p) \vee \uparrow i_j : p \mid 1 \le j < m(p)\}.$$

The variant of mapping FR for monosigned CNF formulas takes the same signature as mapping FR for regular CNF formulas. Given a monosigned CNF formula Γ and a many-valued variable p occurring in Γ whose domain is $N_\Gamma(p) = \{i_1, \ldots, i_{m(p)}\}$ and $i_1 \le i_2 \le \cdots \le i_{m(p)}$ under the order $\le$ associated with N, each positive monosigned literal occurring in the input monosigned CNF formula of the form $\{i_1\}\!:\!p$ is replaced with $\neg(\uparrow i_2 : p)$, of the form $\{i_{m(p)}\}\!:\!p$ is replaced with $\uparrow i_{m(p)} : p$, and of the form $\{i_j\} : p$, where $1 < j < m(p)$, is replaced with $\uparrow i_j : p \wedge \neg(\uparrow i_{j+1} : p)$; and each negative monosigned literal occurring in the input monosigned CNF formula of the form $\{\overline{i_1}\} : p$ is replaced with $\uparrow i_2 : p$, of the form $\{\overline{i_{m(p)}}\} : p$ is replaced with $\neg(\uparrow i_{m(p)} : p)$, and of the form $\{\overline{i_j}\} : p$, where $1 < j < m(p)$, is replaced with $\neg(\uparrow i_j : p) \vee \uparrow i_{j+1} : p$. In addition, it is added the set of clauses that link regular interpretations with Boolean interpretations as in the regular case.

The problem with mapping FR for monosigned CNF formulas is that the size of the derived formula can be exponential in the size of the input formula. This is so because we must apply distributivity when mapping clauses containing positive monosigned literals.

3.5 Regular mapping (R)

The regular (R) mapping is a new mapping whose complexity is better than the complexity of the previous mappings, and that is inspired by mapping FR.

Given a monosigned CNF formula Γ, the signature of mapping R is $\Sigma = \{\{i\} : p, \uparrow i : p \mid \{i\} : p \text{ or } \{\overline{i}\} : p \text{ occur in } \Gamma\}$; i.e., the Boolean signature of Γ extended with regular signs. The Boolean CNF formula derived by mapping R is formed by (i) the clauses of Γ under Boolean semantics; (ii) the set of clauses of mapping FR that link regular interpretations with Boolean interpretations; and (iii) a set of clauses, for each variable p occurring in Γ, that link monosigned literals with regular literals. Assume that $N_{\Gamma}(p) = \{i_1, i_2, \ldots, i_{m(p)}\}$. Then, we add the following clauses

$$\{\{i_1\}:p \leftrightarrow \neg(\uparrow i_2 : p)\} \cup \{\{i_j\}:p \leftrightarrow \uparrow i_j : p \wedge \neg(\uparrow i_{j+1} : p) \mid 1 < j < m(p)\} \cup$$
$$\{\{i_{m(p)}\}:p \leftrightarrow \uparrow i_{m(p)} : p\}$$

The idea of mapping R is that we maintain the input monosigned CNF formula under Boolean semantics but we use both regular and monosigned literals to link many-valued interpretations with Boolean interpretations. This way we avoid applying distributivity. The size of the SAT instance generated by mapping R from a monosigned CNF formula Γ is in $\mathcal{O}(|\Gamma|)$.[2]

3.6 Half regular mapping (HR)

We now define another mapping, called half regular (HR) mapping, which is between FR and R. We defined R in order to avoid applying distributivity. To this end, R maintains the input monosigned CNF formula under Boolean semantics. Since the blowup is only due to the encoding of positive monosigned literals, HR maps negative monosigned literals as in mapping FR and positive monosigned literals as in mapping R. This way, the size of the SAT instance generated by mapping HR from a monosigned CNF formula Γ is also in $\mathcal{O}(|\Gamma|)$.

4 Experimental Investigation

We next report the experimental investigation we conducted to evaluate the performance of the mappings on a number of benchmarks: graph coloring, random binary CSPs, pigeon hole, and all interval series. All the experiments were performed with PC's Pentium III with 1.1 Ghz under Linux, and the SAT solvers used were Chaff and Siege_v4.

parameters			S			FR			HR			R			FL			L		
n	p	k	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%
400	0.02	3	494	335	80	606	186	68	523	194	60	670	504	66	556	183	92	441	176	72
200	0.13	5	518	208	66	726	555	76	603	472	72	445	157	60	1052	1207	56	1214	1214	2
80	0.5	13	137	9	84	61	4	88	69	6.45	88	111	4.2	84	4.4	2.4	98	65	17.17	96
70	0.5	8	228	82	78	116	12	98	177	20	98	330	36	92	255	75	98	424	94	46
60	0.5	8	284	101	58	173	30	84	313	54	90	238	368	76	200	60	92	902	631	48
50	0.5	8	418	125	52	436	132	88	413	212	92	490	133	62	501	231	90	548	117	66

Table 1. Experimental results for graph coloring with Chaff. Time in seconds.

In the first experiment, we considered flat graph coloring problems, generated with the generator of Culberson [12]. The parameters of the generator are: number of vertices (n), number of colors (k), and edge density (p). We created a sample formed by 6 sets of 50 instances; the number of variables (n) ranges from 50 to 400, the number of colors (k) ranges from 3 to 8 and the edge density (p) ranges from 0.01 to 0.5. The parameter settings were designed to sample across the phase transition following the recommendations given by Culberson.[3] Table 1 shows the experimental results obtained: for each set we give the percentage of instances solved (%) using a cutoff of 5000 seconds, and the mean (m) and median (md) time of the solved instances. The best performing mapping is FL, and then FR, HR and R; and the worst performing are L and S.

[2] Observe that all the added clauses have at most three literals, and the number of added clauses is in $\mathcal{O}(lit)$, where lit is the number of occurrences of distinct literals occurring in Γ. Since $|\Gamma| \geq lit$, the size of the instance generated by HR is in $\mathcal{O}(|\Gamma|)$.

[3] http://web.cs.ualberta.ca/~joe/Coloring/Generators/settings.html

In the second experiment, we considered SAT-encoded random binary CSPs using the direct encoding [25]. We used a publicly available generator of uniform random binary CSPs[4] —designed and implemented by Frost, Bessière, Dechter and Regin— that implements the so-called model B: in the class $\langle n, d, p_1, p_2 \rangle$ with n variables of domain size d, we choose a random subset of exactly $p_1 n(n-1)/2$ constraints (rounded to the nearest integer), each with exactly $p_2 d^2$ conflicts (rounded to the nearest integer); p_1 may be thought of as the *density* of the problem and p_2 as the *tightness* of constraints. We incorporated into the generator the automatic generation of all the classes of SAT encodings, and created a representative sample of instances of the hard region of the phase transition described in [24] that could be solved within a reasonable time. The sample is formed by 9 sets of 100 instances; the number of variables ranges from 15 to 70, the domain size was selected in such a way that the instances could be solved within a reasonable time, the density was set at values greater than 0.3 in order to avoid sparse constraint problems, and the tightness was derived from the remaining parameters using the equation $p_2 = 1 - m^{\frac{-2}{p_1(n-1)}}$ in order to generate instances of the hard region of the phase transition [24]. The experimental results obtained are shown in Table 2. We used a cutoff of 2500 seconds. The first column contains the parameters given to the generator of random binary CSPs. The best performing mappings are FR and HR, and then mapping R, and the worst performing are S, FL, and L.

parameters	S			FR			HR			R			FL			L		
$\langle n, d, p_1, p_2 \rangle$	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%
$\langle 15, 25, \frac{80}{105}, \frac{283}{625} \rangle$	23	31	100	18	21	100	20	23	100	22	26	100	117	109	100	23	28	100
$\langle 15, 30, \frac{80}{105}, \frac{424}{900} \rangle$	94	102	100	52	60	100	54	69	100	79	94	100	448	428	100	87	103	100
$\langle 25, 15, \frac{198}{300}, \frac{65}{225} \rangle$	254	236	100	77	73	100	86	80	100	229	207	100	514	502	100	1022	884	100
$\langle 25, 20, \frac{198}{300}, \frac{126}{400} \rangle$	329	208	56	504	470	96	477	523	96	437	397	60	415	452	34	85	82	52
$\langle 35, 10, \frac{305}{595}, \frac{23}{100} \rangle$	116	96	100	38	35	100	43	39	100	96	82	100	145	132	100	147	121	100
$\langle 35, 15, \frac{305}{595}, \frac{60}{225} \rangle$	106	88	12	564	623	44	511	479	42	229	192	16	653	653	4	155	146	14
$\langle 40, 8, \frac{400}{780}, \frac{12}{64} \rangle$	46	39	100	16	15	100	18	17	100	44	39	100	46	44	100	66	59	100
$\langle 45, 10, \frac{415}{990}, \frac{22}{100} \rangle$	587	649	78	428	386	100	451	372	100	594	619	84	646	717	88	560	520	70
$\langle 70, 5, \frac{880}{2415}, \frac{3}{25} \rangle$	10	8.5	100	6	5	100	7.5	6.5	100	4	8	100	9	8.5	100	21	19	100

Table 2. Experimental results for Random Binary CSPs with Chaff. Time in seconds.

holes	S	FR	HR	R	FL	L
9	2.3	0.6	0.6	80.25	4	2
10	21	3	8	540	12	204
11	466	86	34	1230	172	3000
12	3040	150	220	2140	940	1114
13	> 5000	3600	872	> 5000	3890	> 5000
14	> 5000	> 5000	> 5000	> 5000	> 5000	> 5000

Table 3. Experimental results for the pigeon hole problem with Chaff. Time in seconds.

In the third experiment, whose results are shown in Table 3, we studied the scaling behavior of the mappings on pigeon hole instances, where the number of holes ranges from 9 to 14. We used a cutoff of 5000 seconds. The best performing mapping is HR, and then FR and FL, and the worst performing are S, R and L.

In the fourth experiment, whose results are shown in Table 4, we studied the scaling behavior of the mappings on all interval series instances, where the size of the vector ranges from 9 to 17.

[4] http://www.lirmm.fr/~bessiere/generator.html

$\lvert v \rvert$	S	R	HR	L
9	0.01	0	0.02	0.38
11	2.5	0.07	2.47	280
13	1066	47.51	185.58	1878
15	> 5000	527.85	> 5000	> 5000
17	> 5000	> 5000	> 5000	> 5000

Table 4. Experimental results for the all interval series problem with Chaff. Time in seconds.

We used a cutoff of 5000 seconds. The best performing mapping is R, and then HR, and the worst performing are L and S.

We can conclude that mapping S, which is commonly found in SAT repositories, is not the best option, and it is worth exploring alternative encodings. On the one hand, mappings FL and FR are the best for the first two problems but mapping HR has a very good behaviour on average. On the other hand, mapping HR has a linear complexity and does not need to apply distributivity; that fact leads to a poor performance of mappings FL and FR on some problems because of the size of the derived formula.

We believe that the good performance is due to the fact that Boolean variables of regular and logarithmic encodings capture subsets of elements of the domain which are not captured when dealing with the Boolean monosigned signature. This leads to learn shorter clauses; for example, on the hardest binary CSP and coloring instances, the learned clauses by Chaff with mapping HR are between two and three times shorter than the learned clauses by Chaff with mapping S.

parameters			S			FR			HR			R			FL			L		
n	p	k	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%
400	0.02	3	468	136	96	284	46	100	520	91	98	476	94	100	411	135	96	286	58	96
200	0.13	5	32	7	100	22	10	100	25	9	100	25	5	100	2358	2220	4	2783	2600	18
50	0.5	8	13	2	100	37	23	100	46	8	100	23	3	100	63	16	100	9	2	100

Table 5. Experimental results for graph coloring with Siege_v4. Time in seconds.

parameters	S			FR			HR			R			FL			L		
$\langle n, d, p_1, p_2 \rangle$	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%	m	md	%
$\langle 25, 20, \frac{198}{300}, \frac{126}{400} \rangle$	1596	1427	90	1124	909	100	1320	919	96	1004	717	100	1445	1390	20	1265	846	90
$\langle 35, 15, \frac{305}{595}, \frac{60}{225} \rangle$	2907	3395	40	2367	2303	74	2457	2366	48	2122	1880	56	> 5000	> 5000	0	3156	3539	32
$\langle 45, 10, \frac{415}{990}, \frac{22}{100} \rangle$	841	630	100	402	336	100	430	355	100	410	340	100	1638	1416	96	1081	845	100

Table 6. Experimental results for Random Binary CSPs with siege_v4. Time in seconds.

Finally, in order to see if a similar behaviour is observed with other SAT solvers, we repeated the above experiments with Siege_v4. The experimental results obtained are shown in Tables 5–8. In all the experiments we used a cutoff of 5000 seconds. For random binary CSPs and graph coloring instances we only report the results of the hardest instances for Chaff. From the tables, we can conclude that mapping S is not generally the best option and it is worth to try the others mappings we have defined when solving SAT-encoded combinatorial problems with Siege_v4. For the graph coloring instances we have tested we observe that FL is not as good as it was for Chaff, and we do not see many differences among the other encodings. For the random binary CSPs instances, we observe a behaviour similar to Chaff: mappings FR, HR and R allow us to solve more instances with our cutoff. For the pigeon hole instances, the best mapping is FL, but mapping HR, which is

holes	S	FR	HR	R	FL	L
9	63	2.14	2.46	2.59	15	6.56
10	289	10	8.75	9	18	56
11	> 5000	30	51	170	49	238
12	> 5000	162	246	196	74	> 5000
13	> 5000	> 5000	533	> 5000	345	> 5000
14	> 5000	> 5000	> 5000	> 5000	1460	> 5000

Table 7. Experimental results for the pigeon hole problem with Siege_v4. Time in seconds.

$\|v\|$	S	R	HR	L
9	0.06	0.04	0.01	0.03
11	0.87	1.36	0.41	2.05
13	3.96	0.75	2.98	0.01
15	59	22	127	12
17	> 5000	375	> 5000	> 5000

Table 8. Experimental results for the all interval series problem with Siege_v4. Time in seconds.

the best mapping for Chaff, also scales nicely. For the all interval series instances mapping R is, like in Chaff, the best option.

References

1. T. Alsinet, R. Béjar, A. Cabiscol, C. Fernández, and F. Manyà. Minimal and redundant SAT encodings for the all-interval-series problem. In *Proceedings of the Catalan Conference on Artificial Intelligence, CCIA 2002, Castellón, Spain*, pages 139–144. Springer LNCS 2504, 2002.
2. C. Ansótegui, R. Béjar, A. Cabiscol, C. M. Li, and F. Manyà. Resolution methods for many-valued CNF formulas. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing, SAT-2002, Cincinnati, USA*, pages 156–163, 2002.
3. C. Ansótegui, J. Larrubia, C. M. Li, and F. Manyà. Mv-Satz: A SAT solver for many-valued clausal forms. In *4th International Conference Journées de L'Informatique Messine, JIM-2003, Metz, France*, 2003.
4. C. Ansótegui, J. Larrubia, and F. Manyà. Boosting Chaff's performance by incorporating CSP heuristics. In *9th International Conference on Principles and Practice of Constraint Programming, CP-2003, Kinsale, Ireland*, pages 96–107. Springer LNCS 2833, 2003.
5. C. Ansótegui, F. Manyà, R. Béjar, and C. Gomes. Solving many-valued SAT encodings with local search. In *Proceedings of the Workshop on Probabilistics Approaches in Search, 18th National Conference on Artificial Intelligence, AAAI-2002, Edmonton, Canada, 2002*, 2002.
6. M. Baaz and C. G. Fermüller. Resolution-based theorem proving for many-valued logics. *Journal of Symbolic Computation*, 19:353–391, 1995.
7. B. Beckert, R. Hähnle, and F. Manyà. Transformations between signed and classical clause logic. In *Proceedings, International Symposium on Multiple-Valued Logics, ISMVL'99, Freiburg, Germany*, pages 248–255. IEEE Press, Los Alamitos, 1999.
8. B. Beckert, R. Hähnle, and F. Manyà. The SAT problem of signed CNF formulas. In D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht, 2000.
9. R. Béjar, A. Cabiscol, C. Fernández, F. Manyà, and C. P. Gomes. Capturing structure with satisfiability. In *7th International Conference on Principles and Practice of Constraint Programming, CP-2001,Paphos, Cyprus*, pages 137–152. Springer LNCS 2239, 2001.
10. R. Béjar, R. Hähnle, and F. Manyà. A modular reduction of regular logic to classical logic. In *Proceedings, 31st International Symposium on Multiple-Valued Logics (ISMVL), Warsaw, Poland*, pages 221–226. IEEE CS Press, Los Alamitos, 2001.
11. R. Béjar and F. Manyà. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proceedings of the 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'99, London, England*, pages 22–31. Springer LNAI 1638, 1999.

12. J. Culberson. Graph coloring page: The flat graph generator. See http://web.cs.ualberta.ca/~joe/Coloring/Generators/flat.html, 1995.
13. G. Escalada-Imaz and F. Manyà. The satisfiability problem for multiple-valued Horn formulæ. In *Proceedings, International Symposium on Multiple-Valued Logics, ISMVL'94, Boston/MA, USA*, pages 250–256. IEEE Press, Los Alamitos, 1994.
14. A. M. Frisch and T. J. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-2001*, pages 282–288, 2001.
15. R. Hähnle. Towards an efficient tableau proof procedure for multiple-valued logics. In *Selected Papers from Computer Science Logic (CSL'90), Heidelberg, Germany*, LNCS 533, pages 248–260. Springer, 1991.
16. R. Hähnle. *Automated Deduction in Multiple-Valued Logics*, volume 10 of *International Series of Monographs in Computer Science*. Oxford University Press, 1994.
17. R. Hähnle. Advanced many-valued logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 2. Kluwer, second edition, 2001.
18. F. Manyà. *Proof Procedures for Multiple-Valued Propositional Logics.* PhD thesis, Universitat Autònoma de Barcelona, 1996.
19. F. Manyà. The 2-SAT problem in signed CNF formulas. *Multiple-Valued Logic. An International Journal*, 5(4):307–325, 2000.
20. F. Manyà, R. Béjar, and G. Escalada-Imaz. The satisfiability problem in regular CNF-formulas. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, 2(3):116–123, 1998.
21. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*, 2001.
22. N. V. Murray and E. Rosenthal. Resolution and path-dissolution in multiple-valued logics. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems, ISMIS'91, Charlotte, NC*, pages 570–579. Springer LNAI 542, 1991.
23. S. D. Prestwich. Local search on SAT-encoded colouring problems. In *Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing*, pages 105–109. Springer LNCS 2919, 2003.
24. B. Smith and M. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.
25. T. Walsh. SAT v CSP. In *Proceedings of the 6th International Conference on Principles of Constraint Programming, CP-2000, Singapore*, pages 441–456. Springer LNCS 1894, 2000.