Capturing Structure with Satisfiability

Ramón Béjar[1], Alba Cabiscol[2], Cèsar Fernàndez[2], Felip Manyà[2], and Carla Gomes[1]

[1] Dept. of Comp. Science, Cornell University, Ithaca, NY 14853, USA
{bejar,gomes}@cs.cornell.edu
[2] Dept. of Comp. Science, Universitat de Lleida, Jaume II 69, 25001 Lleida, Spain
{alba,cesar,felip}@eup.udl.es

Abstract. We present Regular-SAT, an extension of Boolean Satisfiability based on a class of many-valued CNF formulas. Regular-SAT shares many properties with Boolean SAT, which allows us to generalize some of the best known SAT results and apply them to Regular-SAT. In addition, Regular-SAT has a number of advantages over Boolean SAT. Most importantly, it produces more compact encodings that capture problem structure more naturally. Furthermore, its simplicity allows us to develop Regular-SAT solvers that are competitive with SAT and CSP procedures. We present a detailed performance analysis of Regular-SAT on several benchmark domains. These results show a clear computational advantage of using a Regular-SAT approach over a pure Boolean SAT or CSP approach, at least on the domains under consideration. We therefore believe that an approach based on Regular-SAT provides a compelling intermediate approach between SAT and CSPs, bringing together some of the best features of each paradigm.

1 Introduction

In the last few years, the tremendous advance in the state of the art of SAT solvers, combined with progress in hardware design, has led to the development of very fast SAT solvers. As a consequence, SAT encodings have become competitive with specialized CSP encodings in several domains. However, there is a tradeoff between using a uniform encoding, such as SAT, and a more structured encoding, as found in the CSP paradigm.

In general, CSP-based encodings capture problem structure in a more natural way than SAT encodings. CSP encodings therefore allow in principle for highly efficient solution strategies that exploit inherent problem structure. However, in order to take full advantage of the CSP approach, the user may be required to develop specialized propagation and search techniques that may be difficult to implement efficiently. In a SAT formulation, some of the intricate problem structure may be lost, but the availability of highly optimized general SAT solvers can often compensate for not directly exploiting inherent problem structure.

Our goal is to provide an encoding paradigm that is sufficiently uniform, so that we can develop general solvers, and at the same time allows us to recover the problem structure in a more straightforward manner. Our approach is based

T. Walsh (Ed.): CP 2001, LNCS 2239, pp. 137–152, 2001.

on using so-called Regular-SAT encodings. Regular-SAT retains the uniformity and simplicity of Boolean SAT but in addition captures problem structure in a more straightforward manner. Regular-SAT is an extension of Boolean Satisfiability based on a special class of many-valued CNF formulas, called regular CNF formulas [14,15]. These clausal forms have their origin in the many-valued logic community [3] and are similar to Boolean CNF formulas, except that they use a generalized notion of literal. A literal now is an expression of the form $S : p$, where p is a propositional variable and S is a subset of truth values having a particular structure.

Although more general than SAT, Regular-SAT has many properties in common with traditional Boolean SAT. For example, we have tractable cases such as Regular 2-SAT [18] and Regular Horn-SAT [15], and there exist well-defined phase transitions boundaries in random formula ensembles [5].

Our results show that we can solve certain combinatorial problems more efficiently by using Regular-SAT encodings, compared to approaches based on state-of-the-art SAT or CSP approaches. We present results for both local search and systematic search. Moreover, we show that the Regular-SAT encodings nicely preserve certain structural properties of the original problem domain. In particular, we consider the so-called backbone structure of the problem domain. Phase-transition properties of backbone structure are properly preserved in a Regular-SAT encoding; in a Boolean SAT approach, on the other hand, the phase transition structure in the backbone is not directly recoverable.

The paper is structured as follows. We begin by formally defining the satisfiability problem of Regular CNF formulas (Section 2). In the next section, we describe Regular-DP and Regular-WalkSAT, which are generalizations of the so called Davis-Putnam procedure (though it is actually due to Davis, Logemann and Loveland [9]) and WalkSAT [22]. In Section 4, we present a detailed evaluation of the performance of Regular-SAT procedures. In Section 5, we compare Boolean SAT and Regular SAT w.r.t. capturing problem structure. Section 6 gives overall conclusions.

2 The SAT Problem of Regular CNF Formulas

Regular-SAT is the problem of deciding the satisfiability of regular CNF formulas. A *regular CNF formula* is a classical propositional conjunctive clause form based on a generalized notion of literal, called *regular literal*. Given a truth value set T ($|T| \geq 2$) equipped with a total ordering $\leq$, a *regular literal* is an expression of the form $S : p$, where p is a propositional variable and S is a subset of T which is either of the form $\uparrow i = \{j \in T \mid j \geq i\}$ or of the form $\downarrow i = \{j \in T \mid j \leq i\}$ for some $i \in T$. The informal meaning of $S : p$ is "p is constrained to the values in S", and one can consider the language of regular CNF formulas as a constraint programming language between SAT and CSP.

Definition 1. *A* truth value set *is a non-empty set* $T = \{i_1, i_2, \ldots, i_n\}$, *equipped with a total ordering* $\leq$. *A* sign *is a set* $S \subseteq T$ *of truth values. For each element* i *of the truth value set* T, *let* $\uparrow i$ *denote the sign* $\{j \in T \mid j \geq i\}$,

and let $\downarrow i$ *denote the sign* $\{j \in T \mid j \leq i\}$. *A sign* S *is* regular *if it is identical to* $\uparrow i$ *or to* $\downarrow i$ *for some* $i \in T$.

Definition 2. *A* regular literal *is an expression of the form* $S : p$, *where* S *is a regular sign and* p *is a propositional variable. The complementary literal of* $S : p$ *is* $(T \setminus S) : p$. *A regular literal* $S : p$ *is of* positive *(*negative*)* polarity *if* S *is of the form* $\uparrow i$ *(*$\downarrow i$*) for some* $i \in T$. *A* regular clause *is a finite set of regular literals. A* regular CNF formula *is a finite set of regular clauses.*

Example 1. Let T be the set $\{0, 1, 2\}$ with the standard order on natural numbers. An example of regular CNF formula is

$$(\downarrow 0 : p_1 \vee \downarrow 1 : p_2 \vee \uparrow 2 : p_3) \wedge (\uparrow 1 : p_1 \vee \downarrow 0 : p_2).$$

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 regular literal* $S : p$ *iff* $I(p) \in S$. *An interpretation satisfies a regular clause iff it satisfies at least one of its regular literals. A regular CNF formula* Γ *is* satisfiable *iff there exists at least one interpretation that satisfies all the regular clauses in* Γ. *A regular CNF formula that is not satisfiable is* unsatisfiable. *The empty regular clause, denoted by* $\Box$, *is always unsatisfiable and the empty regular CNF formula is always satisfiable.*

Regular-SAT has advantages over SAT, as well as interesting computational properties:

- SAT is a special case of Regular-SAT: any SAT instance can be transformed into a logically equivalent Regular-SAT instance of the same size by taking $T = \{0, 1\}$ and replacing every literal $p \quad (\neg p)$ with $\uparrow 1 : p \quad (\downarrow 0 : p)$.
- Regular CNF formulas are a more expressive representation formalism than classical CNF formulas, and give rise to more compact encodings (less clauses, variables, etc.) for many combinatorial problems.
- Classical proof methods like resolution, and satisfiability algorithms like Davis-Putnam, GSAT and WalkSAT can be generalized to deal with regular CNF formulas in a natural way. As we will see, the good properties of the classical algorithms remain in regular algorithms, and one does not have to start from scratch when designing algorithms and heuristics.
- Regular-SAT, like SAT, is one of the syntactically and conceptually simplest NP-complete problems. The design, implementation and analysis of algorithms for Regular-SAT tend to be easier than for other CSP algorithms.
- Using regular signs instead of arbitrary subset of truth values as signs has clear advantages. For instance, 2-SAT is solvable in polynomial-time when signs are regular while it is NP-complete for arbitrary signs [18]. Horn CNF formulas admit a natural generalization because regular signs have polarity and Regular Horn-SAT is solvable in polynomial-time.

3 Regular-SAT Algorithms

In this section we first describe Regular-DP and then Regular-WalkSAT, which are generalizations of the Davis-Putnam procedure and WalkSAT. Regular-DP is based on the following rules:

Regular one-literal rule: given a regular CNF formula Γ containing a regular unit clause $\{S:p\}$,

1. remove all clauses containing a literal subsumed by $\{S:p\}$; i.e., all clauses containing a literal $S':p$ such that $S \subseteq S'$;
2. delete all occurrences of literals $S'':p$ such that $S \cap S'' = \emptyset$.

Regular branching rule: reduce the problem of determining whether a regular CNF formula Γ is satisfiable to the problem of determining whether $\Gamma \cup \{S:p\}$ is satisfiable or $\Gamma \cup \{(T \setminus S):p\}$ is satisfiable, where $S:p$ is a regular literal occurring in Γ and the regular literal $(T \setminus S):p$ is its complement.

The pseudo-code of Regular-DP is shown in Figure 1. It returns true (false) if the input regular CNF formula Γ is satisfiable (unsatisfiable). First, it applies repeatedly the regular one-literal rule and derives a simplified formula Γ'. Once the formula cannot be further simplified, it selects a regular literal $S:p$ of Γ', applies the branching rule and solves recursively the problem of deciding whether $\Gamma' \cup \{S:p\}$ is satisfiable or $\Gamma' \cup \{(T \setminus S):p\}$ is satisfiable. In the pseudo-code, $\Gamma_{S:p}$ denotes the formula obtained after applying the regular one-literal rule to a regular CNF formula Γ using the regular unit clause $\{S:p\}$.

Observe that the Davis-Putnam procedure is a particular case of Regular-DP. Our implementation of Regular-DP incorporates two branching heuristics which are extensions of the two-sided Jeroslow-Wang rule [15,6]. Given a regular CNF formula Γ, such heuristics select a regular literal L occurring in Γ that

procedure Regular-DP

Input: a regular CNF formula Γ
Output: `true` if Γ is satisfiable and `false` if Γ is unsatisfiable
`begin`
`if` $\Gamma = \emptyset$ `then return true`;
`if` $\square \in \Gamma$ `then return false`;
/* regular one-literal rule*/
`if` Γ contains a unit clause $\{S':p\}$ `then` Regular-DP($\Gamma_{S':p}$);
let $S:p$ be a regular literal occurring in Γ;
/* regular branching rule */
`if` Regular-DP($\Gamma_{S:p}$) `then return true`;
`else return` Regular-DP($\Gamma_{(T \setminus S):p}$);
`end`

Fig. 1. The Regular-DP procedure

maximizes $J(L) + J(\overline{L})$, where $J(L)$ can be defined either as in Equation 1 or as in Equation 2:

$$J(L) = \sum_{\substack{\exists L' : L' \subseteq L \\ L' \in C \in \Gamma}} 2^{-|C|} \quad (1) \qquad J(L) = \sum_{\substack{\exists L' : L' \subseteq L \\ L' \in C \in \Gamma}} \left(\prod_{S : p \in C} \frac{|T| - |S|}{2(|T| - 1)} \right) \quad (2)$$

where $\overline{L}$ denotes the complement of literal L, $L' \subseteq L$ denotes that literal L' subsumes literal L, $|C|$ denotes the number of literals in clause C, and $|S|$ the number of truth values in sign S.

Equation 1 assigns a larger value to those regular literals L subsumed by regular literals L' that appear in many small clauses. This way, when Regular-DP branches on $\overline{L}$, the probability of deriving new regular unit clauses is larger. Equation 2, that was used in our experiments, takes into account the length of regular signs as well. This fact is important because regular literals with small signs have a larger probability of being eliminated during the application of the regular one-literal rule. Observe that in the case that $|T| = 2$ we get the same equation.

Regular-WalkSAT, whose pseudo-code is shown in Figure 2, tries to find a satisfying interpretation for a regular CNF formula Γ performing a greedily biased walk through the space of possible interpretations. It starts with a randomly generated interpretation I. If I does not satisfy Γ, it proceeds as follows: (i) it randomly chooses an unsatisfied clause C, (ii) it chooses — using function select-WalkSAT — a variable-value pair (p', k') from the set S of pairs (p, k) such that C is satisfied by the current interpretation I if the truth value that I assigns to p is changed to k, and (iii) it creates a new interpretation I' that is identical to I except that $I'(p') = k'$. Such changes are repeated until either

```
procedure Regular-WalkSAT

Input: a regular CNF formula Γ, MaxChanges, MaxTries and ω
Output: a satisfying interpretation of Γ, if found
 begin
  for i := 1 to MaxTries
    I := a randomly generated interpretation for Γ;
    for j := 1 to MaxChanges
      if I satisfies Γ then return I;
      Pick one unsatisfied clause C from Γ;
      S := { (p, k) | S' : p ∈ C, k ∈ S' };
      (p', k') := select-WalkSAT( S, Γ, ω );
      I := I with the truth assignment of p' changed to k';
  return "no satisfying interpretation found";
 end
```

Fig. 2. The Regular-WalkSAT procedure

a satisfying interpretation is found or a pre-set maximum number of changes (MaxChanges) is reached. This process is repeated as needed, up to a maximum of MaxTries times.

Function select-WalkSAT calculates, for each pair $(p, k) \in S$, the number of broken clauses; i.e. the number of clauses that are satisfied by I but that would become unsatisfied if the assignment of p is changed to k. If the minimum number of broken clauses found (u) is greater than zero then either it randomly chooses, with probability ω, a pair (p', k') from S or it randomly chooses, with probability $1 - \omega$, a pair (p', k') from those pairs for which the number of broken clauses is u. If $u = 0$, then it randomly chooses a pair from those pairs for which $u = 0$.

To our best knowledge, the first implementations of local search algorithms for non-Boolean satisfiability were Regular-GSAT [6] and Regular-WalkSAT [7]. In our experiments we used the last available version (10.0) of Regular-WalkSAT, which is faster than the previous ones. Recently, Frisch and Peugeniez [10] have considered a class of non-Boolean formulas where the signs of literals are singletons, and have implemented an efficient local search algorithm for this kind of formulas. Their results show that using non-Boolean satisfiability encodings and solvers is a competitive generic problem solving approach. The reader is invited to consult [3] for related many-valued satisfiability problems and algorithms.

4 Performance Evaluation

A key question regarding Regular-WalkSAT and Regular-DP is how their performance compares to standard WalkSAT and DP. In this section, we consider four benchmark domains. Our results show that there is a concrete computational advantage to using the Regular-SAT procedures, at least on these domains. This suggests that the more compact Regular-SAT encodings, which also preserve more of the problem structure, allow for a more efficient, yet general, solution strategy.

This section is divided in two parts. The first part summarizes our results on three benchmarks, graph coloring, round robin scheduling, and all interval series. These results were obtained as part of the first author's Ph.D. dissertation [4]. Here we only present a summary of the main results. We refer to the thesis for a detailed description of the problem encodings and more detailed run time data.

The second part gives a more detailed evaluation of our Regular-WalkSAT strategy on the quasigroup domain, which provides a structured benchmark with fine control of problem hardness.

4.1 Graph Coloring, Round Robin, and All Interval Series

Our problem domains are graph coloring (flat graphs [8] and DIMACS benchmark instances), round robin scheduling [19], and all interval series (`ais`) [16]. The problems were selected not because of their inherent hardness per se, but because they are known to be hard to solve with SAT algorithms.

For local search algorithms, we observed that the mean cost needed to solve an instance with Regular-WalkSAT is smaller than with WalkSAT in the three

problems. This was true in terms of both number of flips and time, although the difference in the time needed was not as significant as in the number of flips[1]. It was shown in [4] that the performance for the round robin problem is slightly better with Regular-WalkSAT, and is considerably better for the other two problems.

Figure 3 shows the mean number of flips and mean time needed to solve instances of the `ais` problem of different size. The number of flips varies from 7 times to 10 times smaller with Regular-WalkSAT and the time is always about 2 times smaller with Regular-WalkSAT.

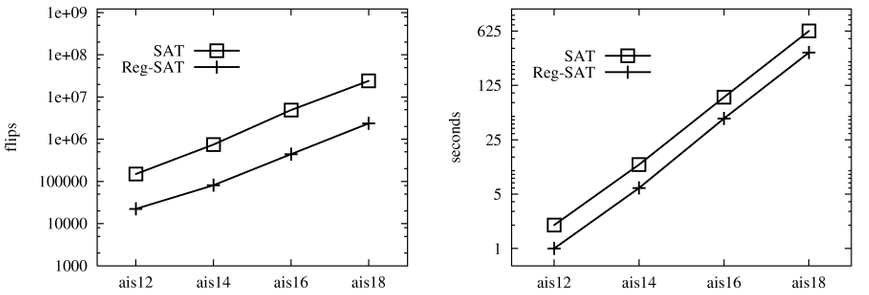

Fig. 3. Scaling behaviour of Regular-SAT and SAT on the `ais` problem

Local search algorithms for Regular-SAT are better in terms of the mean cost, but also in terms of the cost distribution as a whole. In fact, the cost distribution for Regular-SAT on a particular instance dominates the cost distribution for SAT on the same instance. In other words, the probability of finding a solution in less than x flips is always greater with Regular-WalkSAT for each x. Moreover, we observed that the computational cost follows an exponential distribution, at least when solving the instances with approximately optimal noise. This was observed before for local search algorithms for SAT [16]. Figure 4 shows the distribution for the number of flips (RLD) for both algorithms when solving the DIMACS graph coloring instance `DSJC125.5.col`. The figure also shows the exponential distributions (EDs) that were found to best approximate the empirical distributions. The expression for the cumulative form of the EDs is $ed[m](x) = 1 - 2^{-x/m}$, where m is the median of the distribution. The approximations were derived using the Marquart-Levenberg algorithm.

For systematic search, we compared the performance of Regular-DP with the performance of DP when solving Regular-SAT and SAT encodings, respectively, of flat graph problem instances. When we say DP we mean our implementation of Regular-DP but working with $T = \{0, 1\}$. In order to study only the benefits of the encodings, both algorithms used the function of Equation 2 in the branching heuristic. Table 1 shows the mean cost needed to solve a flat graph instance

[1] However, we cannot consider our current version of Regular-WalkSAT (10.0) as optimized as the current one of WalkSAT (35.0).

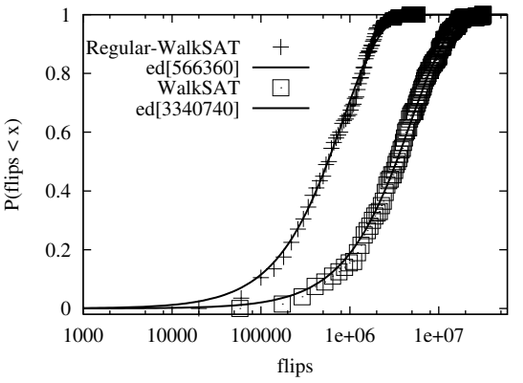

Fig. 4. RLDs for Regular-SAT and SAT on instance `DSJC125.5.col`

with both approaches, as well as the coefficient of variation (CV) that is the ratio between the standard deviation and the mean of the cost. The table shows results for sets of instances obtained with different values for the number of vertices and the number of colors used. For 4 colours and 150 vertices, only 10% of instances were solved with DP, and 85% of instances were solved with Regular-DP; in both cases we used a cutoff of 4 hours, and the results shown correspond to the instances successfully solved by both approaches. Observe that even if the number of nodes in Regular-DP is not smaller in all the cases, the time is always smaller. The likely explanation for this phenomenon is that the number of unit propagations per node is sufficiently small to compensate for a larger number of backtrack nodes. These results indicate that Regular-DP, using our simple branching heuristic, is more effective on the Regular-SAT encoding. In fact, the information contained in the regular literals may help the heuristic to make better decisions. We expect that by incorporating more sophisticated heuristics in Regular-DP (e.g. extensions of look-ahead [17] and look-back [2] heuristics) we will extend the range and size of instances that Regular-DP can solve faster than state-of-the-art SAT solvers.

4.2 Quasigroup Domain

The quasigroup with holes problem (QWH) was recently introduced in [1]. This problem considers randomly generated instances of the quasigroup (or Latin square) completion problem (QCP) [13], and all instances are satisfiable and thus well-suited for evaluating local search methods. The structure of QWH is similar to that found in real-world domains; for example timetabling, routing, and scheduling. Instances are generated by first randomly generating a complete quasigroup, and then erasing some of the colors of the quasigroup (punching "holes"). The hardness of completing a QWH instance can be finely controlled by the number of holes punched. With relatively few holes, a completion is easy because the problem is highly constrained; similarly, instances with a large frac-

Table 1. Results of DP and Regular-DP on flat graph instances

colors		vertices = 100 Encoding				vertices = 150 Encoding			
		classical		regular		classical		regular	
		mean	CV	mean	CV	mean	CV	mean	CV
3	nodes	349	1.05	89	0.88	6182	1.71	597	1.12
	time (sec)	0.70	1.04	0.075	0.90	19	1.57	0.80	1.05
4	nodes	133572	1.96	156303	1.97	2457689	1.5	3861096	1.95
	time (sec)	470	1.91	191	1.88	9955	1.3	5920	1.83

tion of holes are relatively easy to solve, since the instances are under-constrained and many possible completions exists. In [1], it is shown that there is a region of very hard completion problems in between these two extremes. The hard instances arise in the vicinity of a phase transition threshold in the average size of the so-called backbone [1]. The backbone of an instance measures the amount of shared structure among solutions [20,1]. In Section 5, we show that Regular-SAT captures the backbone in a natural way.

Encoding. We have encoded this problem using similar SAT and Regular-SAT encoding schemas. In the SAT encoding, each variable represents a color assigned to a particular cell, so if n is the order (or size) of the quasigroup, we have n^3 variables (n^2 cells with n colors each). Then, we generate clauses that encode the following constraints:

1. Some color must be assigned to each cell.
2. No color is assigned to two cells in the same row.
3. No color is assigned to two cells in the same column.

The first constraint generates clauses of length n with positive literals, and the second and third ones generate binary clauses with negative literals. The total number of clauses generated is $O(n^4)$.

In the Regular-SAT encoding, each variable represents a cell of the quasigroup and the truth value assigned to it represents the color of the cell, so we have $O(n^2)$ variables and n truth values. Then, we generate clauses that encode the same constraints as in the SAT encoding, except for the first constraint. This constraint does not need to be stated explicitly in the Regular-SAT encoding, because a many-valued interpretation to the variables of the formula ensures that each cell receives exactly one color. For encoding the constraint that a particular color i cannot be assigned to two different cells $c1$ and $c2$ of the same row (or column) we generate a regular clause of the form

$$(\downarrow i-1 : c_1 \vee \uparrow i+1 : c_1 \vee \downarrow i-1 : c_2 \vee \uparrow i+1 : c_2).$$

By repeating this clause for all the possible colors, we ensure that $c1$ and $c2$ do not receive the same color. The total number of clauses generated is also $O(n^4)$.

Local search results. In order to compare the typical performance of the Boolean SAT with the Regular-SAT approach, we solved hard QWH instances (*i.e.*, at the phase transition boundary in the backbone) of different orders. For each order, we considered 100 instances and solved the SAT and Regular-SAT encodings using WalkSAT [22] and Regular-WalkSAT [4], respectively. Every instance was solved 100 times with both algorithms. The implementation of WalkSAT used is the one available in the SATLIB and the implementation of Regular-WalkSAT is the one used in [4] (implemented in C++).

Table 2. Median cost for SAT, Regular-SAT and CSP when solving hard QWH instances of different order (at the phase transition)

order	flips		time (seconds)		
	SAT	Regular-SAT	SAT	Regular-SAT	CSP
27	964,849	168,455	2.1	1.5	1.7
30	2,985,105	525,884	7.2	4.8	6.9
33	11,123,065	1,520,667	27.1	16.2	57.1
36	30,972,407	5,099,701	70.8	53.9	1422.3

Table 2 shows the median cost, in time and flips, of all the test-sets used. The cost for a particular instance is defined as the mean time and mean number of flips needed to find a solution. We have also included results for the median time when using a CSP-based systematic search algorithm implemented with the constraint programming library ILOG and that uses the all-different constraint and the *R-brelaz-R* randomized branching strategy [12,21,23]. The results show that the median cost is smaller for the Regular-SAT approach, although between SAT and Regular-SAT the difference is more significant in terms of the number of flips. The greater difference in the number of flips can be in part attributed to the fact that the Regular-SAT encoding is more compact in terms of the number of variables. However, this difference does not directly translate in an equivalent difference in overall run time because the flip rate (flips per second) in Regular-WalkSAT is lower than in WalkSAT. At least some of this difference can be attributed to a higher level of optimization of the WalkSAT code. Despite that our implementation of Regular-WalkSAT is not so optimized, Table 2 still shows that Regular-WalkSAT also outperforms the other approaches in overall run time.

Figure 5 shows graphically the scaling bevahior in time and flips when we increase the order of the QWH instances. We see that the relative good performance of Regular-SAT scales up nicely with the order of the quasigroup. These results are consistent with the experimental results obtained with the other problem domains tested in [4] and summarized in Section 4.1.

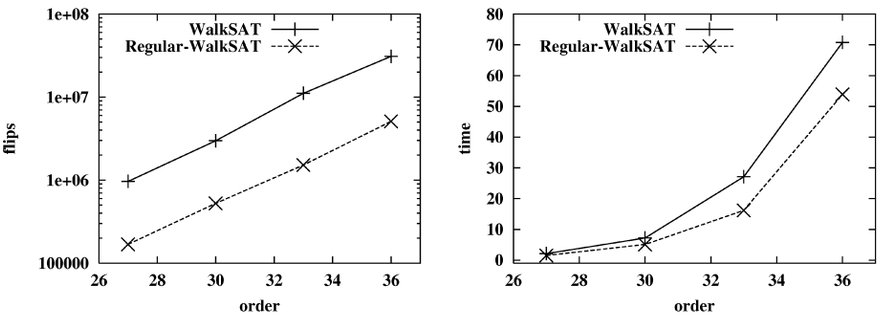

Fig. 5. Scaling behaviour of the median hardness for Regular-SAT and SAT on QWH instances

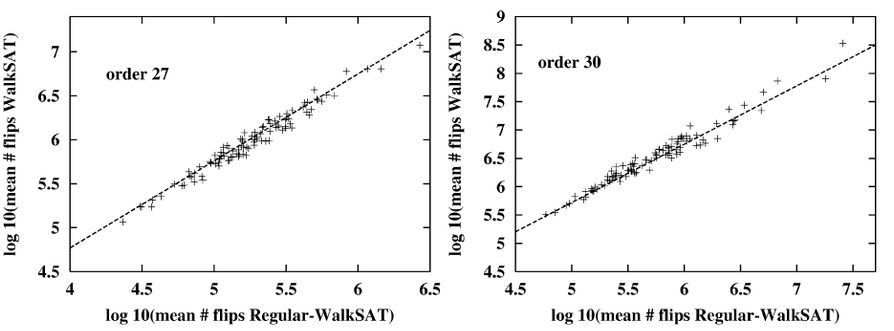

Fig. 6. Correlation between mean cost with Regular-SAT and SAT for order 27 ($a = 0.99, b = 0.81$ and $R_a^2 = 0.97$) and order 30 ($a = 1.03, b = 0.57$ and $R_a^2 = 0.96$).

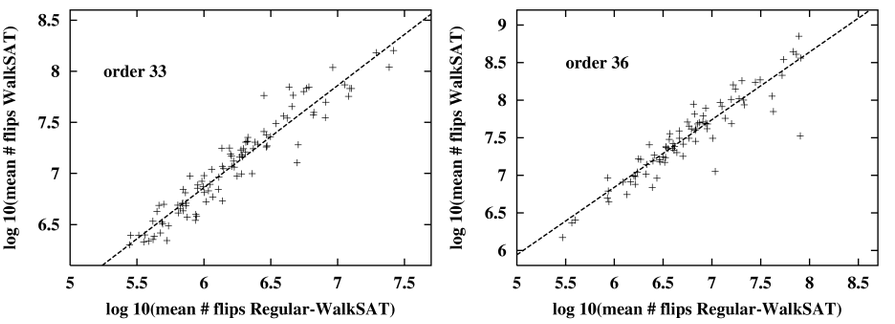

Fig. 7. Correlation between mean cost with Regular-SAT and SAT for order 33 ($a = 1.00, b = 0.86$ and $R_a^2 = 0.91$) and order 36 ($a = 0.90, b = 1.44$ and $R_a^2 = 0.86$)

We have also performed a regression analysis to study the relation between the computational cost of the two different approaches for all the instances of a given test-set. This kind of analysis allows us to investigate to what extent the superior performance observed for the median instance is also observed for any randomly obtained instance within the test-set. Figure 6 (left) shows the results of the regression analysis performed with the instances of order 27. A least-mean-squares (lms) regression analysis of the logarithm of the cost was performed. The figure shows the scatter plot, where each data point (x, y) represents the logarithm, in base 10, of the mean number of flips performed by Regular-WalkSAT (x value) and the same quantity for WalkSAT (y value) when solving a particular instance. The figure also shows the linear equation obtained by the regression analysis ($log_{10}(y) = a \log_{10}(x) + b$) and the adjusted coefficient of determination (R_a^2) that quantifies to what extent the model obtained fits the experimental data. Observe that by working with the logarithm of the data the actual functional relation we are fitting is $y = (10^b) \cdot x^a$. We see that, for order 27, a is close to 1 while 10^b is about 6.45. So, the relative performance increase for Regular-SAT holds uniformly for both easy, medium, and hard instances within the test-set.

Figures 6 (right) and 7 give the correlation analysis results for orders 30, 33, and 36. Observe that the fit of the experimental data is better for the smaller orders. A possible explanation is that as the order increases, the variability in the hardness of QWH instances increases. To properly model the correlation between instance hardness and relative performance may require a more complex regression model. Nevertheless, our analysis still suggests that the increase in performance of Regular-SAT holds fairly uniformly across each test-set.

Although the average complexity of solving instances from a problem domain distribution gives us a valuable information about the difficulty of the problem, the complexity of solving individual instances obtained with the same parameters can vary drastically from instance to instance. So, a more detailed analysis requires a study of the complexity of solving individual instances. To do so, we have constructed empirical Run-time distributions (RTDs) and Run-length (number of flips needed) distributions (RLDs) for both local search algorithms when solving the same instance. The methodology followed has been the one used in [16]. We have focused our attention on the median instance and the hardest instance of a given test-set. Here we present results for the test-set of quasigroups of order 33. Figure 8 shows the RLDs and RTDs for Regular-SAT and SAT on the median instance and also the RTD for CSP on the same instance. These empirical RLDs, in the cumulative form shown, give the probability that the algorithm finds a solution for the instance in less than the number of flips of the $x-$axis (similarly in the RTDs). We observe that Regular-SAT strictly dominates SAT; *i.e.*, the probability of finding a solution with Regular-SAT in less than x flips is always greater than the probability of finding a solution with SAT. Regular-SAT dominates the CSP approach even more significantly than SAT in the run time. Figure 9 shows the same results but for the hardest instance of the same test-set. We observe a similar relative difference between the run time performance of the three approaches.

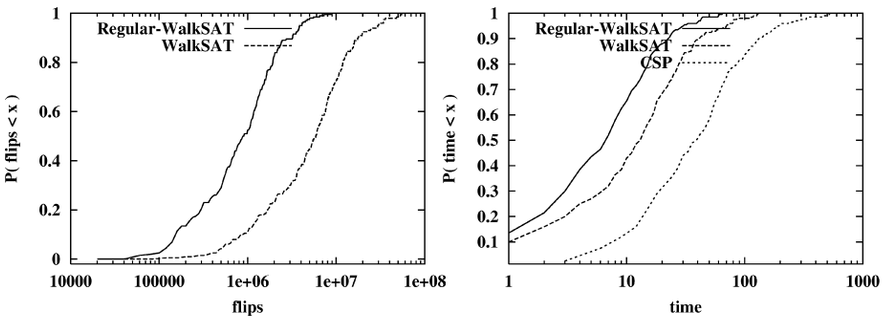

Fig. 8. RLDs (left) and RTDs (right) on the median instance for order 33

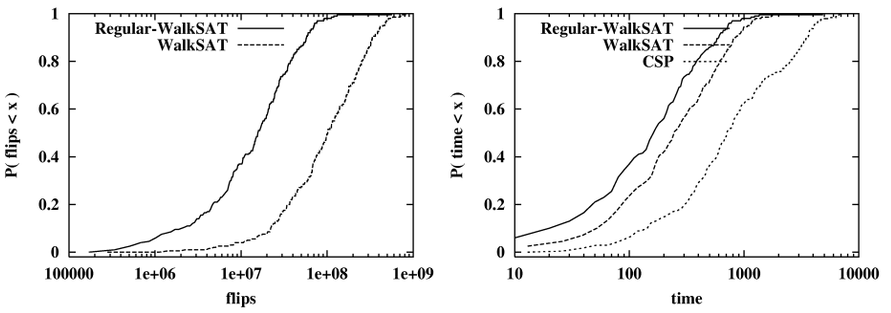

Fig. 9. RLDs (left) and RTDs (right) on the hardest instance for order 33

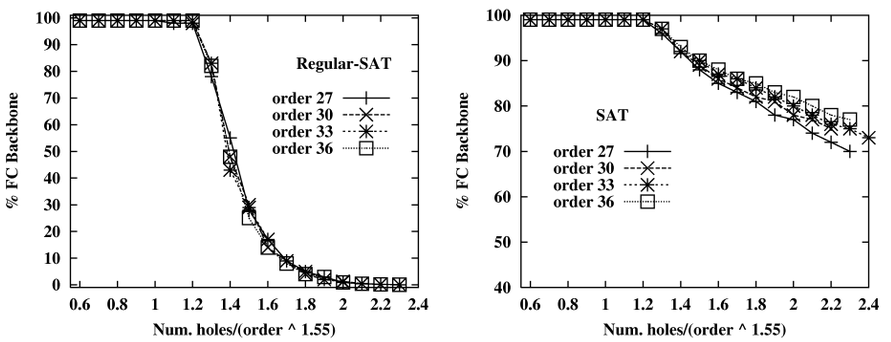

Fig. 10. The average forward-checing backbone for Regular-SAT (left) and SAT (right) on QWH instances

5 The Backbone Structure

We now consider the structure of the backbone in the QWH problem. Informally speaking, the backbone measures the amount of shared structure among the set of all solutions to a given problem instance [20]. The size of the backbone is measured in terms of the percentage of variables that have the same value in all solutions. Achlioptas *et al.* [1] observed a transition from a phase where the size of the backbone is almost 100% to a phase with a backbone size close to 0%. The transition is sudden and coincides with the hardest problem instances both for incomplete and complete search methods.

For efficiency purposes, Achlioptas *et al.* also propose a slightly weaker version of the backbone, which is computed by only using forward-checking (FC) to find shared variable settings in the solution set. They show that this backbone is qualitatively similar to the original notion of backbone. We adapted the notion of SAT FC backbone for Regular-SAT, which is obtained by applying the one-literal rule to every regular literal of the formula and computing the fraction of the total number of variables that becomes constrained to a single truth value.

The left panel of Figure 10 shows the FC backbone for QWH instances of different orders and with a different number of holes for the Regular-SAT encoding. We observe a phase transition in the fraction of backbone variables for the Regular-SAT encoding. In contrast, the right panel of Figure 10 displays the FC backbone structure for the Boolean SAT encoding. As we see from the figure, the SAT encoding does *not* properly preserve the phase transition properties of the backbone structure.[2] The Regular-SAT encoding can capture a structural property such as the backbone more faithfully than the Boolean SAT encoding.

6 Conclusions

We have shown that Regular-SAT provides an attractive approach for encoding and solving combinatorial problems. The formulation provides an intermediate alternative to the SAT and CSP approaches, and combines many of the good properties of each paradigm. Its similarity to SAT allows us to extend existing SAT algorithms to Regular-SAT without incurring excessive overhead in terms of computational cost. We have shown, using a range of benchmark problems, that Regular-SAT offers practical computational advantages for solving combinatorial problems. In addition, Regular-SAT maintains more of the original problem structure compared to Boolean SAT encodings. By providing more powerful search heuristics and optimizing the data structures, we expect to further extend the reach of the Regular-SAT approach.

Acknowledgements. We would like to thank Bart Selman for useful comments and discussions that helped to improve the paper. This research was partially

[2] This phenomenon was initially observed by [1,11]. One can still recover the phase transition of the backbone for the SAT encoding by restricting the backbone count to include only the variables set positively, as done in [1].

funded by the DARPA contracts F30602-00-2-0530 and F30602-00-2-0596, and project CICYT TIC96-1038-C04-03. The fourth author was supported by the "Secretaría de Estado de Educación y Universidades". This research was also partially funded by the Intelligent Information Systems Institute, Cornell University, funded by AFRL/AFOSR (F49620-01-1).

References

1. D. Achlioptas, C. P. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instances. In *Proc. of AAAI-2000*, pages 256–261, 2000.
2. R. J. Bayardo and R. C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. of AAAI'97*, pages 203–208, 1997.
3. B. Beckert, R. Hähnle and F. Manyà. The SAT problem of signed CNF formulas. In *Labelled Deduction*, pages 61–82, Kluwer, 2000.
4. R. Béjar. *Systematic and Local Search Algorithms for Regular-SAT*. PhD thesis, Universitat Autònoma de Barcelona, 2000.
5. R. Béjar and F. Manyà. Phase transitions in the regular random 3-SAT problem. In *Proc. of ISMIS'99*, pages 292–300. LNAI 1609, 1999.
6. R. Béjar and F. Manyà. A comparison of systematic and local search algorithms for regular CNF formulas. In *Proc. of ECSQARU'99*, pages 22–31. LNAI 1638, 1999.
7. R. Béjar and F. Manyà. Solving combinatorial problems with regular local search algorithms. In *Proc. of LPAR'99*, pages 33–43. LNAI 1705, 1999.
8. J. C. Culberson and F. Luo. Exploring the k-colorable landscape with iterated greedy. In *Cliques, Coloring and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. 1996.
9. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.
10. A.M. Frisch and T.J. Peugniez. Solving non-Boolean satisfiability problems with stochastic local search. In *Proc. of IJCAI'01*, pages 282–288, 2001.
11. C. Gomes, H. Kautz, and Y. Ruan. QWH - A structured benchmark domain for local search. *Technical Report, Intelligent Information Systems Institute (IISI), Cornell University*, 2001.
12. C. Gomes, B. Selman, and N. Crato. Heavy-tailed distributions in combinatorial search. In *Proc. of CP'97*, pages 121–135. Springer LNCS 1330, 1997.
13. C. P. Gomes and B. Selman. Problem structure in the presence of perturbations. In *Proc. of AAAI'97*, pages 221–226, 1997.
14. R. Hähnle. Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation*, 4(6):905–927, 1994.
15. R. Hähnle. Exploiting data dependencies in many-valued logics. *Journal of Applied Non-Classical Logics*, 6:49–69, 1996.
16. H. Hoos and T. Stützle. Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning*, 24(4):421–481, 2000.
17. C. M. Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Proc. of CP'97*, pages 341–355. LNCS 1330, 1997.
18. F. Manyà. The 2-SAT problem in signed CNF formulas. *Multiple-Valued Logic. An International Journal*, 5(4):307–325, 2000.
19. K. McAloon, C. Tretkoff, and G. Wetzel. Sports league scheduling. In *Proc. of ILOG International Users Meeting*, 1997.

20. R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(8), 1999.
21. J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of AAAI'94*, pages 362–367, 1994.
22. B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proc. of AAAI'94*, pages 337–343, 1994.
23. K. Stergiou and T. Walsh. The difference all-difference makes. In *Proc. of IJCAI'99*, 1999.