# Modeling Choices in Quasigroup Completion: SAT vs. CSP [*] [†]

**Carlos Ansótegui**
Computer Science Department
Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
carlos@eup.udl.es

**Alvaro del Val** and **Iván Dotú**
Escuela Politécnica Superior
Universidad Autónoma de Madrid
28049 Madrid, Spain
{delval, ivan.dotu}@ii.uam.es

**Cèsar Fernández** and **Felip Manyà**
Computer Science Department
Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
{cesar, felip}@eup.udl.es

## Abstract

We perform a systematic comparison of SAT and CSP models for a challenging combinatorial problem, quasigroup completion (QCP). Our empirical results clearly indicate the superiority of the 3D SAT encoding (Kautz *et al.* 2001), with various solvers, over other SAT and CSP models. We propose a partial explanation of the observed performance. Analytically, we focus on the relative conciseness of the 3D model and the pruning power of unit propagation. Empirically, the focus is on the role of the unit-propagation heuristic of the best performing solver, Satz (Li & Anbulagan 1997), which proves crucial to its success, and results in a significant improvement in scalability when imported into the CSP solvers. Our results strongly suggest that SAT encodings of permutation problems (Hnich, Smith, & Walsh 2004) may well prove quite competitive in other domains, in particular when compared with the currently preferred channeling CSP models.

## Introduction

The Quasigroup Completion Problem (QCP) is a very challenging benchmark among combinatorial problems, which has been the focus of much recent interest in the area of constraint programming (Gomes & Shmoys 2002b; Dotú, del Val, & Cebrián 2003). It has a broad range of practical applications such as conflict-free wavelength routing in wide band optical networks, statistical design, and error correcting codes (Gomes & Shmoys 2002b); it has been put forward as a benchmark which can bridge the gap between purely random instances and highly structured problems (Gomes & Shmoys 2002a); and its structure as a multiple permutation problem (Walsh 2001; Hnich, Smith, & Walsh 2004) is common to many other important problems in constraint satisfaction. Thus, solutions that prove effective on QCPs have a good chance of being useful in other problems with similar structure.

Motivated by earlier results by (Gomes & Shmoys 2002b; 2002a) and (Dotú, del Val, & Cebrián 2003) with integer programming and CSP models, we perform a systematic study of modeling choices for quasigroup completion, testing a variety of solvers and heuristics on various SAT and CSP encodings. The clear winner is the SAT 3D-encoding proposed in (Kautz *et al.* 2001), specially with the solver Satz (Li & Anbulagan 1997), closely followed by the solver Satzoo (Eén & Sörensson 2003) on the same encoding. As these two solvers are quite different (one uses a strong form of lookahead in its heuristic, but no backjumping or learning, while the other relies heavily on the last two), the 3D encoding appears to be quite robust as a representation. On the other hand, CSP models perform significantly worse with the two solvers we tried, and standard SAT encodings generated from the CSP models are simply too large in practice. These results strongly suggest that the 3D encoding can turn out to be quite competitive in other permutation problems (many of which arise in quite practical problems (Hnich, Smith, & Walsh 2004)) when compared with the currently preferred channeling models.

The reasons for this appear to be twofold. First, we can show that the 3D encoding (which is basically the "SAT channeling model" of (Hnich, Smith, & Walsh 2004) extended to multiple permutations and dual models) exactly captures the channeling models of QCPs as defined e.g. in (Dotú, del Val, & Cebrián 2003), but in a much more concise way, by collapsing primal and dual variables. Further, we can show that the 3D encoding captures the "support SAT encoding" of the channeling model, hence by results of (Gent 2002), that unit propagation on the 3D encoding achieves the same pruning as arc consistency (MAC) in the CSP channeling model. These results appear easy to extrapolate to other permutation problems (or similar ones with "channeling constraints"), which have received a lot of recent attention (Cheng *et al.* 1999; Walsh 2001; Hnich, Smith, & Walsh 2004). Second, empirically, we identify Satz's UP heuristic as crucial to its success in this domain; as shown by the fact that, when importing the heuristic into our CSP solvers, we obtain significant improvements in their scalability. Further, the improvements are much smaller if we only use lookahead to detect potential wipeouts (i.e. for "failed literal detection"), but choose variables instead by some other standard heuristic such as min-domain.

The structure of this paper is as follows. In sections 2 and

[*]We thank Carla Gomes for the quasigroup generator lsencode and for computational resources to run the experiments, C. Bessiére for the MAC code, and the authors of all solvers.

[†]This research has been partially supported by projects TIC2001-1577-C03-03 and TIC2000-0539 of the Ministerio de Ciencia y Tecnologia, Spain.

3 we describe the QCP and various SAT and CSP encodings. We then present in Section 4 the main empirical results for someone who just wants to know the fastest way to solve this kind of problem. Section 5 analyzes the theoretical relationships between representations along the lines discussed above on conciseness and propagation power. Sections 6 and 7 inquire into solver-specific features that may explain the success of Satz. As described above, the focus is on Satz's UP heuristic and its incorporation into CSP solvers. Finally, we list some open questions and directions for future work.

## The Quasigroup Completion Problem

A quasigroup is an ordered pair $(Q, \cdot)$, where $Q$ is a set and $\cdot$ is a binary operation on $Q$ such that the equations $a \cdot x = b$ and $y \cdot a = b$ are uniquely solvable for every pair of elements $a, b$ in $Q$ (Gomes & Shmoys 2002b). The order n of the quasigroup is the cardinality of the set $Q$. A quasigroup can be seen as an $n \times n$ multiplication table which defines a Latin Square, i.e. a matrix which must be filled with "colors" (the elements of the set $Q$) so that the colors of each row are all distinct, and similarly for columns. We focus on the quasigroup completion problem (QCP), which is the NP-complete problem (Colbourn ) of coloring a partially filled Latin square. QCP share with many real world problems a significant degree of structure, while at the same time allowing the systematic generation of difficult problems by randomly filling the quasigroup with preassigned colors. It is thus ideally suited as a testbed for constraint satisfaction algorithms (Gomes & Shmoys 2002a). Experimental studies of the problem have confirmed its interest for research, by for example helping to discover important patterns in problem difficulty such as heavy-tailed behavior (Gomes *et al.* 2000).

Among the kind of structure that has been identified in many constraint satisfaction problems, and which is shared by QCPs, is that of permutation problems. These are constraint satisfaction (sub)problems with the same number of variables as values, where a solution is a permutation of the values (Walsh 2001). Each row and column of a Latin Square defines a permutation problem, thus the QCP is a multiple permutation problem with $2n$ intersecting permutation constraints ($n$ row permutation constraints and $n$ column permutation constraints).

## SAT and CSP Encodings

The two SAT encodings of the QCP of order $n$ considered in this paper, introduced in (Kautz *et al.* 2001), use $n$ Boolean variables per cell; each variable represents a color assigned to a cell, and the total number of variables is $n^3$. The most basic SAT encoding, which is known as 2-dimensional (2-D) encoding, includes clauses that represent the following constraints:

1. at least one color must be assigned to each cell (ALO-1);
2. no color is repeated in the same row (AMO-2); and
3. no color is repeated in the same column (AMO-3).

The other encoding, which is known as 3-dimensional (3-D) encoding, adds to the 2-D encoding redundant clauses that represent the following constraints:

1. each color must appear at least once in each row (ALO-2);
2. each color must appear at least once in each column (ALO-3); and
3. no two colors are assigned to the same cell (AMO-1).

Both encodings have $\mathcal{O}(n^4)$ clauses. The labels associated to each clause set are explained later.

For the sake of brevity and clarity, the only CSP encoding we describe is the "bichanneling model" of (Dotú, del Val, & Cebrián 2003). It consists of:

- A set of *primal variables* $X = \{x_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n\}$; the value of $x_{ij}$ is the color assigned to the cell in the $i$th row and $j$th column, and $n$ is the order of the quasigroup, i.e. the number of rows and columns.
- Two sets of *dual variables*: $R = \{r_{ik} \mid 1 \leq i \leq n, 1 \leq k \leq n\}$, where the value of $r_{ik}$ is the column $j$ where color $k$ occurs in row $i$; and $C = \{c_{jk} \mid 1 \leq j \leq n, 0 \leq k \leq n\}$ where the value of $c_{jk}$ represents the row $i$ where color $k$ occurs in column $j$.

The domain of all variables is $\{1, \ldots, n\}$, where these values represent respectively colors, columns, and rows. Variables of different types are linked by channeling constraints:

- *Row channeling constraints* link the primal variables with the row dual variables: $x_{ij} = k \Leftrightarrow r_{ik} = j$.
- *Column channeling constraints* link the primal variables with the column dual variables: $x_{ij} = k \Leftrightarrow c_{jk} = i$.

As pointed out in (Dotú, del Val, & Cebrián 2003), this model is a complete model of the problem; in particular, the so called primal constraints, which explicitly state that no two colors can be repeated in any one row or column, are redundant, and hinder propagation. CSP "channeling" encodings for permutation problems similar to the one presented here are discussed at length in (Hnich, Smith, & Walsh 2004).

## Experimental results: Part 1

We considered four state-of-the art SAT solvers: Satz (Li & Anbulagan 1997), Chaff (Moskewicz *et al.* 2001), Berkmin (Goldberg & Novikov 2002), and Satzoo (Eén & Sörensson 2003). We chose Satz because some authors have claimed that it is the best option to solve QCPs; our experimental results provide evidence of this claim too. We chose Chaff and Satzoo because they were the winners of the two last SAT competitions, and Berkmin because it is often competitive with Chaff. In addition, we tested two CSP solvers, the GAC library described in (van Beek & Chen 1999), and the MAC solver by Regin and Bessiere (Bessière & Régin 1996). Note that for binary CSPs they are simply different implementations of the MAC algorithm.

The instances tested in our experiments are of the QWH type (quasigroup with holes (Kautz *et al.* 2001), generated with `lsencode`), are all satisfiable, and are located near the phase transition.[1] Our samples, with 200 instances each,

[1]Specifically, QWH instances of order $n$ are generated with $\lceil 1.6 \times n^{1.55} \rceil$ holes.

| | % solved | | | | mean | | | | median | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| order | Chaff | Berkmin | Satzoo | Satz | Chaff | Berkmin | Satzoo | Satz | Chaff | Berkmin | Satzoo | Satz |
| 35 | 99 | 100 | 100 | 100 | 59 | 24 | 37 | 6 | 3.2 | 0.5 | 16 | 1.2 |
| 37 | 96 | 99 | 100 | 100 | 232 | 173 | 129 | 42 | 24 | 4.7 | 5 | 5.8 |
| 40 | 82 | 86 | 96 | 99.5 | 518 | 590 | 861 | 539 | 288 | 112 | 142 | 41 |
| 43 | 50.5 | 62 | 78.5 | 84 | 279 | 1487 | 1799 | 1243 | 1085 | 2178 | 815 | 358 |
| 45 | 46 | 46 | 59.5 | 68 | 380 | 1312 | 1021 | 1181 | $> 12000$ | $> 12000$ | 1857 | 1184 |

Table 1: Comparison of Chaff, Berkmin, Satzoo and Satz on the 3-D encoding. Time in seconds, mean of solved instances. Cutoff 12000 seconds. 1 GHz.

| | % solved | | mean (solved) | | median | |
|---|---|---|---|---|---|---|
| order | Satz | GACvo | Satz | GACvo | Satz | GACvo |
| 35 | 100 | 37 | 6 | 2970 | 1.2 | $> 12000$ |
| 37 | 100 | 11 | 42 | 2572 | 5.8 | $> 12000$ |
| 40 | 99.5 | 7 | 539 | 4546 | 41 | $> 12000$ |

Table 2: Comparison of Satz on the 3-D encoding and GAC-vo. Time in seconds. Cutoff 12000 seconds. 1 Ghz.

did not contain "balanced" problems, which are reported to be the hardest; still, we did verify that they are much harder for all solvers than the balanced problems tested in (Dotú, del Val, & Cebrián 2003), which were not close enough to the phase transition for their class.

Of all the solutions tried, we can discard the 2D SAT encoding and the primal CSP models, as they give significantly worse results. Of the remaining encodings, the 3D encoding was clearly superior with the four SAT solvers, with Satz scaling somewhat better than Satzoo and both much better than chaff. On the other hand, the CSP approach proposed in (Dotú, del Val, & Cebrián 2003), which uses MAC on the bichanneling model, scaled much worse on the (harder) problems tested in this paper. There is therefore a clear dividing line between SAT and CSP encodings in terms of performance. Table 1 provides the data for SAT solvers, and Table 2 compares our best SAT solver with the CSP approach (labeled GACvo) of (Dotú, del Val, & Cebrián 2003), which seems to be the best one in the literature, using GAC on the bichanneling model with a special value ordering heuristic (named `vdom+` in that paper). Note that the ratio of solved problems for GACvo is much lower than reported in (Dotú, del Val, & Cebrián 2003) for problems of the same order, and that's because our instances are harder.[2]

We will now explore potential explanations for these observations, and ways to improve these results for the CSP approaches drawn from these explanations. Our first focus will be on comparing representations; later we consider solver-specific issues (most importantly, the extra level of propagation and the heuristics of Satz).

[2]The improvements reported for the trichanneling model in (Dotú, del Val, & Cebrián 2003) do not affect scaling behavior, and are pretty much subsumed by the stronger forms of lookahead discussed later. Hence we did not test this model for this paper.

## Comparing models

In order to compare our models formally, we need a small detour through *SAT encodings of CSP models*, a third modeling option that we have not considered so far. In this context, the many-valued CSP variables are represented by means of a set of boolean variables, one for each possible assignment. So, for example, a primal variable $x_{ij}$ becomes a set of boolean variables $x_{ij} = 1, \ldots, x_{ij} = n$. The semantics of CSP domains is then captured by including one ALO ("at least one") clause for each variable, specifying that the variable must take at least one of its possible values, and $O(n^2)$ AMO ("at most one") clauses which specify, for every pair of possible values of a variable, that they can not be satisfied simultaneously. In our bichanneling model, we would have variables $x_{ij} = k$, $r_{ik} = j$ and $c_{jk} = i$, for each triple $j, k, i$, for a total of $3n^3$ variables.

Define the *minimal support encoding* of the bichanneling model as that consisting of AMO-ALO clauses for the three variable types, together with the *channeling clauses* $\neg x_{ij} = k \vee r_{ik} = j$ and $\neg r_{ik} = j \vee x_{ij} = k$, which directly encode the equivalence $x_{i1} = j \Leftrightarrow r_{ij} = k$ which defines the constraint between $x_{ij}$ and $r_{ik}$ (and similarly for column channeling constraints). Clearly, the channeling clauses completely characterize the channeling constraints, hence the minimal support encoding is a complete model.

The *support encoding*, as defined in (Gent 2002), encodes a constraint between two variables $X$ and $Y$ by adding, for each possible value $v$ of $X$, the clause $\neg X = v \vee Y = v_1 \vee \ldots \vee Y = v_k$, where $v_1, \ldots, v_k$ is a list of all values $w$ in the domain of $Y$ such that $(v, w)$ satisfies the constraint, i.e. a list of *supports* for the assignment $X = v$. For the bichanneling model, we can observe that the channeling clauses of the minimal support encoding already encode the supports for $x_{ij} = k$ and $r_{ik} = j$. For values $v \neq k$, the supports for $x_{ij} = v$ are given by the clause $\neg x_{ij} = v \vee \bigvee_{h \in \{1,\ldots,n\}, h \neq j} r_{ik} = h$, and for values $v \neq j$, the supports for $r_{ik} = v$ are given by $\neg r_{ik} = v \vee \bigvee_{h \in \{1,\ldots,n\}, h \neq k} x_{ij} = h$.

**Proposition 1** *Unit resolution obtains the same results on the minimal support encoding as in the support encoding of the bichanneling model.*

**Proof:** Consider e.g. the constraint between $x_{i1}$ and $r_{i2}$ (a similar analysis holds for column channeling constraints). Since the channeling clauses are preserved in the minimal encoding, it suffices to show that the effect of unit propagation on the additional support clauses is also obtained without them. For $v \neq 2$ we have the support clause $\neg x_{i1} = v \vee \bigvee_{h \in \{1,\ldots,n\}, h \neq 1} r_{i2} = h$. We consider two cases. First, suppose $r_{i2} = 2, \ldots, r_{i2} = n$ are all false; we must show that unit resolution on the remaining clauses imply $\neg x_{i1} = v$. Unit resolution on the ALO for $r_{i2}$ obtains $r_{i2} = 1$, which together with the channeling clauses yield $x_{i1} = 2$, and with the AMO for $x_{i1}$, $\neg x_{i1} = v$. Second, suppose $x_{i1} = v$ is true and, say, $r_{i2} = 1$ through $r_{i2} = n - 1$ are all false; we need to show that $r_{i2} = n$ follows by unit resolution. Now, $x_{i1} = v$ implies through AMO $\neg x_{i1} = 2$, and thus $\neg r_{i2} = 1$ using the channeling clauses. This together with the hypothesis of the case and the ALO for $r_{i2}$ allows us to obtain $r_{i2} = n$, as desired. ■

Our second observation is that the binary theory consisting of the channeling clauses for all constraints can be simplified using a strongly connected components (SCC) algorithm such as in (del Val 2001). The SCCs will consist precisely of the triplets of boolean variables of the form $x_{ij} = k$, $r_{ik} = j$ and $c_{jk} = i$. The SCC-based algorithm would then replace each such triplet by a single variable, which we may appropriately call $x_{ijk}$, as in the 3D SAT encodings, and perform the appropriate replacements in the remaining clauses. The result is the following:

- The ALO and AMO clauses for the CSP variables become clauses of the 3D-encoding. Specifically, the clauses labeled ALO-k and AMO-k ($1 \leq k \leq 3$) in the 3D-encoding are the result of rewriting the ALO and AMO clauses for the primal variables (k=1), row dual variables (k=2) and column dual variables (k=3) in the minimal channeling encoding.
- The channeling clauses become tautologies after variable replacement, so they can be eliminated.

The SCC-simplification formally captures the intuitive idea that the triplet of variables $x_{ij} = k$, $r_{ik} = j$ and $c_{jk} = i$ all "mean the same" –color $k$ is in cell $(i, j)$–, and hence that each triplet can be "collapsed" into a single boolean variable $x_{ijk}$, as done in the 3D encoding (and in the "SAT channeling model" of (Hnich, Smith, & Walsh 2004)).

**Proposition 2** *Unit resolution on the 3D model has the same pruning power as MAC on the bichanneling model.*

**Proof:** (Gent 2002) shows that unit resolution on the support encoding of a CSP problem is equivalent to MAC on the original problem. We have just shown that unit resolution on the minimal support encoding is equivalent to unit resolution on the support encoding of the bichanneling model, and that the 3D encoding is simply the minimal support encoding after SCC simplification, which does not affect the power of unit resolution. ■

Thus, the 3D model exactly captures the bichanneling model, without loosing any propagation power, but with 3 times fewer variables ($n^3$ instead of $3n^3$) and without the $4n^3$ channeling clauses of the minimal channeling model. We did in fact try to solve QCPs using a direct encoding of the bichanneling model, with very bad results due to the size of the resulting theories. The above propositions seem to go a long way toward explaining the success of the 3D model.

## Satz's heuristic in QCPs

We now turn to solver and domain-specific features that may explain the observed performance. Thus our next step was to analyze in depth the behavior on QCP instances of the best solver, Satz, so as to incorporate new propagation techniques and heuristics into the CSP solvers from the insights gained. Before going into details, let us recall the variable selection heuristic that implements Satz, which combines MOMS (Maximum Occurrences in clauses of Minimum Size) and UP (Unit Propagation) heuristics. In both heuristics, the goal is to maximize the power of unit propagation. MOMS picks one variable among those that occur the most often in minimal size clauses, since these are more likely to result in propagation. UP goes a step further by actually measuring the number of propagations from each choice. It examines each variable $p$ occurring in a given CNF formula $\phi$ by respectively adding the unit clauses $p$ and $\neg p$ to $\phi$, and independently making two unit propagations, which are used to derive a score for each variable. As a secondary effect, UP detects so-called *failed literals* in $\phi$, which when satisfied falsify $\phi$ in a single unit propagation.

```
For each free variable $x$ such that $PROP_z(x)$ is true do
(let $F'$ and $F''$ two copies of the formula $F$ under consideration)
   $F'$ := unit-propagation($F' \cup \{x\}$);
   $F''$ := unit-propagation($F'' \cup \{\neg x\}$);
   If $\Box \in F'$ and $\Box \in F''$ then return "$F$ is unsatisfiable"
   If $\Box \in F'$ then $x$:=0, $F$ := $F''$
   else if $\Box \in F''$ then $x$:=1, $F$ := $F'$
   If $\Box \notin F'$ and $\Box \notin F''$ then
     let $w(x)$ denote the weight of $x$
     $w(x)$ := number of times that non-binary clauses of $F$
       have been reduced when deriving $F'$
     $w(\neg x)$ := number of times that non-binary clauses of $F$
       have been reduced when deriving $F''$
For each free variable $x$ do
   $H(x) := w(x) * w(\neg x) * 1024 + w(x) + w(\neg x)$;
Branch on the free variable $x$ with greatest $H(x)$
```

Figure 1: The variable selection heuristic of *Satz* for $PROP(x, 4)$

In order to reduce the time required to propagate all literals, Satz applies UP to a restricted number of variables that occur in binary clauses by applying a unary predicate, called $PROP_z$, which is defined as follows:

**Definition 1** *Let $\phi$ be a CNF formula; let $PROP(x, i)$ be a binary predicate which is true iff variable $x$ occurs both positively and negatively in binary clauses of $\phi$, and there are at least $i$ occurrences of $x$ in binary clauses of $\phi$; and let $T$ be an integer. $PROP_z(x)$ is defined to be the first of the three predicates $PROP(x, 4), PROP(x, 3), true$ (in this order) whose denotational semantics contains more than a fixed number $T$ of variables. $T$ is set to 10 in Satz.*

After applying unit propagation to a restricted number of variables and detecting failed literals, the heuristic of Satz weights literals with different criteria depending on the predicate applied. For example, when $PROP(x, 4)$ is applied, the heuristic scores each literal $(x, \neg x)$ with the number of times that non-binary clauses have been reduced when propagating the literal. The pseudocode of the variable selec-

| | % solved | | | mean | | | median | | |
|---|---|---|---|---|---|---|---|---|---|
| order | GAC-HLA | MAC-HLA | Satz | GAC-HLA | MAC-HLA | Satz | GAC-HLA | MAC-HLA | Satz |
| 35 | 98 | 98.5 | 100 | 428 | 586 | 6 | 131 | 129 | 1.2 |
| 37 | 86 | 89.5 | 100 | 1360 | 1913 | 42 | 882 | 822 | 5.8 |
| 40 | 52 | 58 | 99.5 | 1770 | 3033 | 539 | 5304 | 8411 | 41 |
| 43 | 30 | 39 | 84 | 1342 | 2668 | 1243 | $> 12000$ | $> 12000$ | 358 |
| 45 | 24 | 26 | 68 | 2810 | 3585 | 1181 | $> 12000$ | $> 12000$ | 1184 |

Table 3: Comparison of GAC-HLA, MAC-HLA and Satz. Time in seconds, mean of solved instances. Cutoff 12000 seconds. 1 Ghz.

| | % solved | | mean | | median | |
|---|---|---|---|---|---|---|
| order | MAC-LA | MAC-HLA | MAC-LA | MAC-HLA | MAC-LA | MAC-HLA |
| 30 | 96 | 100 | 544 | 18 | 482 | 9 |
| 33 | 91 | 100 | 1753 | 187 | 1256 | 52 |
| 35 | 58 | 98.5 | 2714 | 586 | 4487 | 129 |

Table 4: Comparison of MAC-LA and MAC-HLA. Time in seconds, mean of solved instances. Cutoff 12000 seconds. 1 Ghz.

tion heuristic of Satz for $PROP(x, 4)$ is shown in Figure 1. Function $H(x)$ is used to reach a good balance between weights of positive literals and weights of negative literals.

When solving QCP instances using the 3-D encoding with Satz, we observed that Satz almost always uses the predicate $PROP(x, 4)$, which requires $x$ to occur in both positive and negative binary clauses, with at least 4 occurrences in total. A closer look at the 3-D encoding reveals that Boolean variables that fulfill $PROP(x, 4)$ model CSP variables with domain size 2. The reason is that positive literals only occur in the ALO-1, ALO-2 and ALO-3 constraints, hence the only way to have $x$ occur positively in a binary clause is when one of these clauses becomes binary, in which case the corresponding (primal or dual) CSP variable has domain size 2. Further, it is easy to show that the variables in such positive binary clauses have at least three other negative occurrences from AMO clauses, so that $PROP(x, 4)$ holds. This analysis led us to incorporate the technique of failed literals and the heuristic of Satz into CSP solvers as follows:

1. For each free CSP variable of domain size 2, we propagate each value of the domain in order to see if the domain can be reduced. As a result, the domain can remain as before, can be a singleton or can be empty. In the first case, we weight the variable using the balance function $H$ of Satz's heuristics, where $w(x = i)$ is the number of times that domains have been reduced after propagating the value $i$. In the second case, we fix the variable to the only value of its domain. In the third case, we have detected an inconsistency and we backtrack.
2. We select the first free CSP variable of domain size 2 with greatest value of function $H$.
3. If there is no candidate variable in step 2, we apply the default heuristic of the CSP solver (for example, `min-domain`).

The above description corresponds to what we will call simply look-ahead heuristic (LAH). We refer to the version of GAC (MAC) that incorporates LAH as GAC-LAH (MAC-LAH).

## Experimental results: Part 2

To assess the performance of LAH we performed an empirical investigation. In the first experiment we compared Satz with GAC-LAH and MAC-LAH on sets of 200 instances of order 35, 37, 40 and 45 of the hard region of the phase transition. The results obtained are shown in Table 3. We observed that Satz outperforms GAC-LAH and MAC-LAH, but the differences are not so dramatic as with GAC-vo, which was the most competitive CSP option to solve QCPs. MAC-LAH seems to be slightly superior to GAC-LAH.

In the second experiment, we analyzed if the improvements achieved on CSP solvers are due to the use of lookahead to detect potential wipeouts or to the heuristic function. To this end, we compared MAC-LAH with a variant MAC-LA which uses the same lookahead but chooses variables of minimum domain size instead of applying the heuristic function H to a reduced number of variables; we refer to that version as MAC-LA. The results obtained, shown in Table 4, clearly indicate that the heuristic function plays a central role.

Nevertheless, failed literals do achieve an extra-level of consistency in each search node over that of plain unit propagation, and a natural question to ask is whether stronger forms of consistency could yield better results. We experimented with stronger forms of lookahead without success, but did not try `alldifferent` constraints. Hall's Theorem, as presented in (van Hoere 2001), states that: the constraint `alldifferent`$(x_1, \ldots, x_n)$ with respective variable domains $D_1, \ldots, D_n$ has a solution if and only if no subset $K \subseteq \{x_1, \ldots, x_n\}$ exists such that $|K| > |\bigcup_{x_i \in K} D_i|$. However, we observed experimentally that, for the QCP and the solvers considered, the condition of the theorem is only violated by subsets of three CSP variables with domain size two. It is easy to show that the lookahead phase of our heuristic, when applied to a variable of domain size two, finds a contradiction for the two values of the do-

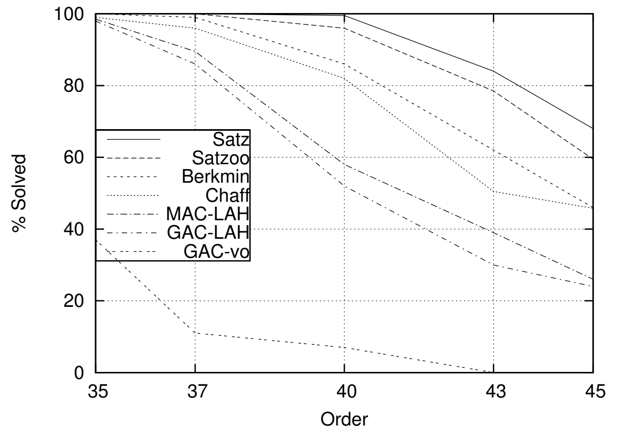

Figure 2: Percent solved for the main solvers.

main and backtracks. So in practice we may be getting the same pruning power as with Hall's theorem.

As a summary of our main results, Figure 2 compares the main approaches discussed in this paper in terms of scalability, plotting percent of solved instances against the order of quasigroups. It can be seen that SAT encodings still scale better, but that the incorporation of the lookahead heuristic inspired by Satz goes a long way toward bridging the gap between the best CSP model available to date, GAC-vo, and the SAT approaches.

## Conclusions and future work

In this paper, we have shown that SAT encodings allow much more scalable solutions to QCP problems, in particular when compared to previous results in the literature. We have explained this performance by properties of the representation and by solver-specific features; and we have shown that those features can also be fruitfully exploited in CSP models to get much better CSP solutions than before.

A number of open questions remain for future work. First, while we can explain Satz's performance and exploit its features in CSP approaches, a similar study could be carried out for Satzoo. This might help in understanding the role of CBJ and learning in QCPs, as in our experience they do not help with QCPs formulated as CSPs. Second, it appears that the single-lookahead heuristic with a MAC-GAC solver may do quite a bit of redundant work, which may already be done when ensuring arc-consistency in the previous search level; exploiting this may lead to a substantially faster implementation of the heuristic. Finally, we plan to study the effect of many-valued models (Ansótegui *et al.* 2002) as an intermediate and potentially more concise representation between SAT and CSP.

## References

Ansótegui, C.; Manyà, F.; Béjar, R.; and Gomes, C. 2002. Solving many-valued SAT encodings with local search. In *Proc. of the Workshop on Probabilistics Approaches in Search, AAAI-2002*.

Bessière, C., and Régin, J. 1996. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *CP-96, 2nd Int. Conf. on Principles and Practice of Constraint Programming*, 61–75.

Cheng, B.; Choi, K.; Lee, J.; and Wu, J. 1999. Constraint propagation by redundant modeling: an experience report. *Constraints* 167–192.

Colbourn, C. 1984. The complexity of completing partial latin squares. *Discrete Applied Mathematics* 25–30.

del Val, A. 2001. Simplifying binary propositional theories into connected components twice as fast. In *LPAR'01, Proc. 8th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning*. LNCS, Springer-Verlag.

Dotú, I.; del Val, A.; and Cebrián, M. 2003. Redundant modeling for the quasigroup completion problem. In *Proc. CP-2003*, 288–302. Springer LNCS 2833.

Eén, N., and Sörensson, N. 2003. Satzoo. http://www.math.chalmers.se/˜een/Satzoo/.

Gent, I. P. 2002. Arc consistency in SAT. In *Proc. ECAI'2002*.

Goldberg, E., and Novikov, Y. 2002. Berkmin: A fast and robust sat-solver. In *Proc. of DATE'02*.

Gomes, C., and Shmoys, D. B. 2002a. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proc. Computational Symposium on Graph Coloring and Extensions*.

Gomes, C., and Shmoys, D. B. 2002b. The promise of LP to boost CSP techniques for combinatorial problems. In *CP-AI-OR'02*, 291–305.

Gomes, C.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning* 24(1/2):67–100.

Hnich, B.; Smith, B. M.; and Walsh, T. 2004. Dual modelling of permutation and injection problems. *Journal of Artificial Intelligence Research*. to appear.

Kautz, H. A.; Ruan, Y.; Achlioptas, D.; Gomes, C. P.; Selman, B.; and Stickel, M. 2001. Balance and filtering in structured satisfiable problems. In *IJCAI'01*, 351–358.

Li, C. M., and Anbulagan. 1997. Look-ahead versus look-back for satisfiability problems. In *CP'97, Proc. Int. Conf. on Principles of Constraint Programming*, 341–355. Springer LNCS 1330.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*.

van Beek, P., and Chen, X. 1999. Cplan: A constraint programming approach to planning. In *AAAI'99*, 585–590.

van Hoere, W. 2001. The alldifferent constraint: A survey. In *Proc. 6th Annual Workshop of the ERCIM Working Group on Constraints, Prague*.

Walsh, T. 2001. Permutation problems and channeling constraints. In *LPAR-2001*.