Encoding Quantified CSPs as Quantified Boolean Formulae

Ian P. Gent and **Peter Nightingale** and **Andrew Rowley**[1]

Abstract. Quantified Constraint Satisfaction Problems (QCSPs) are CSPs in which some variables are universally quantified. For each possible value of such variables, we have to find ways to set the remaining, existentially quantified, variables so that the constraints are all satisfied. Interest in this topic is increasing following recent advances in Quantified Boolean Formulae (QBFs), the analogous generalisation of satisfiability (SAT). We show that we can encode QCSPs as QBFs. We introduce a simple generalisation of the direct encoding of CSPs into SAT. We then introduce some adaptations of this encoding to make it effective in a QBF solver. We solve some QCSP test instances orders of magnitude faster than using a specialised QCSP solver, taking advantage of the more advanced state of the art in QBF solving. Our conclusions are twofold. First, there is considerably more subtlety required in encodings in QBF than in SAT. Second, in an area such as QCSP where algorithmic techniques are not yet highly developed, encodings into a better understood problem can give access to extremely advanced search methods with very little implementation effort.

1 Introduction

A particular advantage of encoding one search problem as another occurs when search techniques for the target problem are more highly developed than the original. This is to be expected when a new search problem starts to be studied intensively. In this paper we consider Quantified Constraint Satisfaction Problems (QCSPs), which has only recently started to be researched [11, 2, 4]. By contrast, a number of solvers are available for Quantified Boolean Formulae (QBFs), exploiting a variety of sophisticated techniques, e.g. conflict-directed [12] and solution-directed backjumping [10]. Therefore, we expect encoding QCSPs into QBFs to be a competitive method of solving QCSPs. Furthermore, we expect it to indicate areas for future of QCSP algorithmic research. In this paper, we show that these two expectations are justified.

A more unexpected result of our research is that the fine tuning of encodings to be effective for search is considerably more involved than in the case of SAT, where encodings often have an elegant simplicity. This issue occupies the bulk of this paper. A simple way of lifting CSP encodings to QCSP is very ineffective, and we need new ideas without analogues in SAT to make search effective. The end result is an encoding which, on the encoded instances, can run many times faster than a QCSP algorithm on the original.

A final issue we address in this paper, which is familiar from other problems such as CSP and QBF, is that of flaws in random instances. We show that flaws arise in the QCSP instance generator proposed by Mamoulis & Stergiou, and indeed quickly infect all generated problems. This is an important problem for experiments in QCSP, but we as yet do not have a solution to it.

2 Background

QCSPs are more expressive than CSPs and in general are PSPACE-complete. They allow formulation of problems where all contingencies must be allowed for, for example where configuration must be possible for all possible sequences of user choices. We briefly describe QCSPs, but refer the reader elsewhere for more details [2, 4]. A binary QCSP is of the form QC where Q is a sequence of quantifiers $Q_1v_1...Q_nv_n$, where each Q_i quantifies ($\exists$ or $\forall$) a variable v_i and each variable occurs exactly once in the sequence Q. C is a conjunction of binary constraints. Each constraint involves two variables v_p, v_q (which may be universally or existentially quantified) and a set of conflicts R. A conflict $c \in R$ is a pair $\langle s_p, s_q\rangle$. If $v_p := s_p$ and $v_q := s_q$ then the constraint is violated. A QCSP of the form $\exists v_1 Q_2 v_2 ... Q_n v_n C$ is true if $Q_2 v_2 ... Q_n v_n C[v_1 := s]$ is true for some value of s. Similarly, a QCSP of the form $\forall v_1 Q_2 v_2 ... Q_n v_n C$ is true if $Q_2 v_2 ... Q_n v_n C[v_1 := s]$ is true for all values of s.

We provide some basic details of QBF, but refer the reader elsewhere for a more detailed introduction [3]. A Quantified Boolean Formula is of the form QC where Q is defined above (however in this case the domain of each variable is T, F). C is a Boolean formula in conjunctive normal form (CNF), a conjunction of clauses where each clause is a disjunction of literals. Each literal is a variable and a sign. The literal is said to be negative if negated and positive otherwise. The semantic definition is the same as for QCSP.

2.1 Assumptions throughout the paper

A common structure in our encodings is the following, where a indicates the truth of the disjunction: $(x_1 \vee x_2 \vee ... \vee x_n) \iff a$. This can be reduced to $n+1$ CNF clauses: $(\neg a \vee x_1 \vee x_2 \vee ... \vee x_n) \wedge \bigwedge_{i=1}^{n} (\neg x_i \vee a)$. Throughout the paper, we present the equivalences in this form for pedagogical purposes, while in implementation we use the clause set. In some cases, the equivalence only applies if a variable z is false, in which case we have that $((x_1 \vee x_2 \vee ... \vee x_n) \iff a) \vee z$. The clauses that arise are the same as before, with the addition of the literal z to each clause.

We present experimental results throughout the paper. Experiments were run on a cluster of computers using Intel Pentium II processors running at 450 MHz, each with 386MB RAM. We used the 'k-QCSP' instance generator [11] with quantifier sequence $\exists\forall\exists$ and $n/3$ variables in each block. We set the p (constraint graph density) parameter to 0.5 and varied q (constraint looseness) and n. We

[1] School of Computer Science, University of St Andrews, St Andrews, Fife, KY16 9SS, UK. `{ipg,pn,agdr}@dcs.st-and.ac.uk`

report runtimes on 100 encoded instances using our own implementation of CBJ for QBF: we found that Solution directed backjumping [10] did not reduce search but caused overheads. We compare against Mamoulis and Stergiou's specialised algorithms for QCSPs [11]. We use the same instances in each case to reduce variation dependent on properties of the instance. We do not include translation times in our results for QBF, but these are not large. The largest average time over a sample was 0.1sec on a 3GHz machine, and this could be improved by more careful implementation. Our results on hard instances taking thousands of seconds would not be affected.

3 Global Acceptability Encoding

In this section we show that the direct encoding of CSP into SAT can be lifted to work for QCSPs, using what we call the global acceptability encoding. The name comes from the use of a single variable to encode acceptability of the QBF assignment to universal variables. We will find that the encoding presented in this section is ineffective for search, but its relative simplicity allows us to focus on the key issues of lifting an encoding correctly from CSP to QCSP, without getting lost in the minutiae of adapting the encoding to be effective.

In the direct encoding, variable v takes value a in the CSP if and only if the variable v_a is true in the SAT instance. Therefore an 'acceptable' assignment of the SAT variables is one where each of the underlying CSP variables has one value. In SAT, it is simple to rule out unacceptable assignments by making the formula false in those cases. In QBF the situation is more complex: if v is existentially quantified, then the Boolean variables $x^v_1 ... x^v_d$ are existentially quantified, and the situation is the same as with SAT. However, if v is universally quantified, for the formula to be true overall requires that it is true for each value of v. Hence the Boolean variables $v_1 ... v_d$ must be universally quantified. The formula should only be falsified overall if an acceptable assignment is false, so unacceptable assignments to universals should cause the formula to be true. To achieve this, we use existentially quantified 'indicator' variables. This follows Gent and Rowley's technique for addressing a similar problem in encoding the game of Connect-4 in QBF [8].

3.1 Encoding the QCSP variables

Each QCSP variable v is represented by a set of Boolean variables $x^v_1 ... x^v_d$ where d is the domain size of v. These variables are quantified existentially or universally depending on the type of the QCSP variable. In either case all QBF variables for v are quantified together, and the order of the original QCSP quantifications is preserved. The intended semantics of the Boolean variables are given by $v \mapsto i \equiv x^v_i \mapsto T$. Not all QBF assignments correspond to anything in the original QCSP, for example those which set a variable to three different values. So we define an "acceptable assignment":

Definition 1 *An acceptable assignment of an encoded QBF is one which corresponds to an assignment of the QCSP. For each QCSP variable v, exactly one of the corresponding QBF variables x^v_i is true. A partial assignment is acceptable if it extends to an acceptable full assignment.*

We encode that the assignment to either existential or universal variables is acceptable. The approaches to doing so are very different: it is easiest to discuss existential variables first. We simply rule out an existential taking no values.

- ALO (at least one value)

$$(x^v_1 \vee x^v_2 \vee ... \vee x^v_n)$$

Two groups of clauses are necessary to set the appropriate indicator variable when the universal assignment is unacceptable. We use indicator variables of three types:

1. l^v indicates (when true) that $\neg(x^v_1 \vee ... \vee x^v_d)$ (i.e. v has no values)
2. $m^v_{i,j}$ indicates that $x^v_i \wedge x^v_j$ (i.e. v has at least two values)
3. z indicates the assignment is unacceptable because some universal variables are set incorrectly. It is true iff one or more of the other indicator variables is true.

- ALO (at least one value)

$$(x^v_1 \vee x^v_2 \vee ... \vee x^v_d) \iff \neg l^v$$

- AMO (at most one value)

$$\bigwedge_{i=1}^{d} \bigwedge_{j=i+1}^{d} \left((\neg x^v_i \vee \neg x^v_j) \iff \neg m^v_{i,j}\right)$$

For any assignment to the universal variables in the QBF, there exists a corresponding assignment to the indicator variables. So the indicator variables are existentially quantified in a block after all other variables. The global indicator variable z is defined with the statements below, where U is the set of all universal QCSP variables.

- Indicator collector clauses:

$$\left[\bigvee_{v \in U} \left(l^v \vee \bigvee_{i=1}^{d} \bigvee_{j=i+1}^{d} m^v_{i,j}\right)\right] \iff z$$

3.2 Encoding the constraints

The clauses encoding the constraints are the same as in the direct encoding from SAT to CSP, via what are called 'conflict clauses' with the addition of the global indicator variable z. This means that all constraint clauses are true if a universal variable is set unacceptably. A constraint C between two variables a and b with domains A and B is represented by the relation $R_{a,b} \subseteq A \times B$ containing the allowed pairs. So $a = i$ and $b = j$ satisfies C if and only if $\langle i, j\rangle \in R_{a,b}$.

- **Conflict clauses** For all tuples $\langle i, j\rangle \notin R$,

$$(\neg x^a_i \vee \neg x^b_j \vee z)$$

With the direct encoding from CSP to SAT, the AMO clauses may be omitted. However, in the quantified case this causes complications. We do omit AMO clauses for existential variables – a valid QBF assignment with n assignments to an existential QCSP variable would represent n valid QBF assignments. However, if multiple assignments to universal variables were allowed, an unacceptable assignment to a universal variable would falsify the formula, so AMO clauses for universal variables are essential to prevent this.

3.3 Correctness and Experimental Analysis

In an acceptable assignment, there is a natural correspondence between assignments of the QBF and the QCSP, so for example $x^v_2 = T$ corresponds to $v = 2$. The other, indicator, variables all depend on the value of the variables x^v_i and are set from them by unit propagation, and when we talk of the QBF assignment corresponding to a QCSP, we assume these assignments of the extra variables.

Theorem 2 *A QCSP is true if and only if the encoded QBF is true, for the global acceptability encoding.*

Proof: We use induction on the unassigned QCSP variables, working from the innermost quantified variables out. The induction hypothesis is that any QCSP under an acceptable partial assignment A with the last k quantified variables unassigned is true iff the encoded QBF under the corresponding acceptable assignment to A is true.

The base is a full assignment to the QCSP, corresponding to a full and acceptable assignment to the QBF. In the full QBF assignment, all indicator variables are false. Thus indicator clauses can all be discarded as all are true. The ALO/AMO clauses can similarly be discarded as all are satisfied in an acceptable assignment. We are left with the constraint clauses excluding the z literal as it is false. But these are all satisfied iff no conflict occurs in the direct encoding. So the QBF is true iff the corresponding assignment satisfies the QCSP.

For the step, we need to show that the result extends one quantification level. First, consider acceptable assignments to the $k+1^{st}$ innermost variable v in the QCSP quantifier. Each acceptable assignment to the corresponding QBF variables has the correct truth value, by the induction hypothesis. We have to show that the QBF is true iff one of these one of these acceptable extensions is true (for the $k+1^{st}$ variable existential) or iff one of them is false (for universal.)

First, consider the $k+1^{st}$ variable v being existential. It must be given at least one value as otherwise an ALO clause will be false. A solution may exist with an unacceptable assignment to the QBF, with more than one value $x_i^v = T$ assigned. But then we just choose (say) the least numbered value, and the acceptable existential assignments corresponding to that value will also satisfy the QBF, as all occurrences of the variable x_i^v in constraint clauses are negative literals.

There remains the case of setting universal QBF variables unacceptably. For any (acceptable or not) setting of universal variables, the existential indicator variables can be set appropriately to satisfy all indicator clauses. In an unacceptable assignment, the indicator clauses are true only when z has to be true, and hence all constraint clauses are satisfied. That is, for any unacceptable setting of universal variables, there is a setting of existentials which makes all clauses true, so the QBF is true in these cases. Since a universal node is false iff one of its children is false, the falsity of the node depends only on the status of acceptable children, as required.

Finally, note that the result applies to the empty assignment of a QCSP, since an empty QBF assignment is acceptable. *QED*

The global acceptability encoding performed dreadfully. For example, at $n = 15, p = 0.5$, median performance was > 1000 sec for all $q > 0.55$ except 0.92. This compares to an absolute maximum of 53.79 sec for FC1 [11], and a largest median of 1.72 sec.

It is not hard to see why the global acceptability encoding performs so badly. One important reason is that propagation is crippled by the necessary presence of the global indicator variable z. Therefore we have no equivalent of the result from SAT, that the algorithm DP performs FC on the direct encoding [6]. For example, consider the conflict clause $(\neg x_1^v \vee \neg x_3^w \vee z)$, setting $x_1^v = T$ fails to set $x_3^w = F$, because the status of z is unknown. This simple propagation fails to take place, yet it is fundamental to making the direct encoding act like forward checking. Worse, z cannot be set false until *all* universal variables are set acceptably. In the next section we will introduce a more complicated way of encoding acceptable assignments, which ameliorates these problems and improves search.

4 Local Acceptability Encoding

The key problem we identified with the global acceptability encoding was the fact that z could not be set until all universal variables were. The solution to this is to use many indicator variables to encode acceptability, one for each universal QCSP variable. Then, as soon any of these is set true, the remainder of the problem is satisfied.

We introduce an indicator variable z_v corresponding to each universal QCSP variable. As before, z_v is existentially quantified in a final block with the other indicator variables. For presenting the encoding, we assume that u, v are consecutive, universally quantified, QCSP variables, possibly with intervening existential variables. In clauses involving u and v, we omit z_u if v is the first universally quantified variable. We first change the collector clauses as follows, the indicator variables c being existential and quantified together with the other indicator variables at the end:

- Indicator collector clauses:

$$(z_u \vee c_v) \iff z_v$$
$$\left(l^u \vee \bigvee_{i=1}^{d} \bigvee_{j=i+1}^{d} m_{i,j}^u \right) \iff c_v$$

Note that this makes unacceptability monotonic, so if u is assigned unacceptably we say that all future universals are. We now adapt other clauses so that once z_v becomes true it makes all clauses true relating to deeper quantification levels.

To save space, we do not show the changes to most clauses, but describe them. For constraint clauses, we change z to z_v as follows: for constraints involving existentials we use z_v for the last universal v quantified before the *first* existential QCSP variable; for constraints involving a universal v we use z_v, or the innermost if the constraint involves two universals.[2] That leaves the ALO/AMO clauses for universals, and we make a critical change so that as soon as a universal variable u is set unacceptably, there is no requirement to set any future universal correctly.[3]

- ALO (at least one value)

$$z_u \vee ((x_1^v \vee x_2^v \vee ... \vee x_d^v) \iff \neg l^v)$$

- AMO (at most one value)

$$\bigwedge_{i=1}^{d} \bigwedge_{j=i+1}^{d} z_u \vee \left((\neg x_i^v \vee \neg x_j^v) \iff \neg m_{i,j}^v\right)$$

Theorem 3 *A QCSP is true if and only if the encoded QBF is true, for the local acceptability encoding.*

Proof: Most of the proof is as before. For example the reader can check that again in a full acceptable assignment, the conflict clauses correctly encode the semantics of the constraints. The difficulty is in the induction step for a universal QCSP variable v. It is no longer the case that all constraint clauses are satisfied by the global z. Instead, we claim that if an unacceptable assignment to the universal leads to falsity, an acceptable assignment does too. No constraint clause involving any x_i^v can be falsified later in the search tree, as it will be satisfied by z_v. Therefore, any search node with a false clause could also be reached by an acceptable assignment of the QBF variables x_i^v and corresponding changes to the indicator variables. This justifies the claim. *QED*

The point of the local acceptability variables is to enable more propagation. In fact, in the case of the direct encoding, unit propagation performs at least as many domain reductions as Mamoulis & Stergiou's generalisation FC0 of FC [11]. FC0 removes values of future variables when they are in conflict with the assigned value of the

[2] In fact a constraint between universal variables renders the problem trivially false if any conflict exists, but we include this case for completeness.

[3] We do not report results here, but found that this change dramatically improved search when added to the rest of the local acceptability encoding.

current variable. This clearly happens from unit propagation in the conflict clauses, because when the outer variable is set, the indicator variable must already be false in an acceptable assignment. So if a variable x_i^a is true, the variable x_j^b must be set false by the conflict clause, equivalent to FC0's removal of the value $b = j$.

Experimentally, the local acceptability encodings provide improved search over the global encodings, but still performs orders of magnitude worse than Mamoulis & Stergiou's techniques. For example, at $n = 15$, $p = 0.5$ and $q = 0.56$, median run time is 53.41 sec compared to 1877 sec for the global encoding, but compared to unmeasurable (< 0.01 sec) median time for FC1.

5 Adapted Log Encoding

In SAT, it has often been noted that just three variables are needed to encode 8 values of a CSP variables, instead of the 8 in the direct encoding [5, 13], reducing the branching factor from 256 to 8. However, since only one value is allowed by ALO/AMO clauses, there is no real reduction, and the use of three variables in clauses for one CSP variable reduces the effect of propagation [13]. For QCSPs we show that the log encoding can be very effective when applied to universal variables only. We show that with subtle adaptations, search on true instances can be greatly improved. We call this the 'adapted log' encoding to distinguish it from the conventional log encoding in SAT.

In the adapted log encoding, we express existential variables as before. For a universal QCSP variable v, we have $\lceil \log_2 d \rceil$ universal QBF variables b_i^v. The setting of these variables corresponds to the values of the QCSP variable in the obvious way (except that we assume QCSP domains start with 1, so we subtract one before encoding into binary.) Unlike the standard log encoding in SAT, we do not use combinations of these QBF variables in constraint clauses. Instead, we introduce new existential variables x_i^v, one for each value of the universal QCSP variable. These are quantified at the end with other indicator variables. For example, if $d = 5$ we enforce that the b's set the x's as follows, if u is the previous universally quantified variable:

- Log value clauses:

$$\begin{array}{l} z_u \vee x_1^v \vee b_2^v \vee b_1^v \vee b_0^v \\ z_u \vee x_2^v \vee b_2^v \vee b_1^v \vee \neg b_0^v \\ z_u \vee x_3^v \vee b_2^v \vee \neg b_1^v \vee b_0^v \\ z_u \vee x_4^v \vee b_2^v \vee \neg b_1^v \vee \neg b_0^v \\ z_u \vee x_5^v \vee \neg b_2^v \vee b_1^v \vee b_0^v \end{array}$$

There is now no need for the ALO/AMO clauses for universal variables. However, there are still unacceptable assignments, when the binary variables suggest a value outside the domain of v. To resolve this we introduce new indicator variables i (as usual existential quantified at the end.) We retain the use of local acceptability variables z_v, to indicate when an illegal value has been given.

- Out of domain indicator clauses:

$$\begin{array}{l} z_u \vee (i_6^v \iff (b_2^v \vee \neg b_1^v \vee b_0^v)) \\ z_u \vee (i_7^v \iff (b_2^v \vee b_1^v \vee \neg b_0^v)) \\ z_u \vee (i_8^v \iff (b_2^v \vee b_1^v \vee b_0^v)) \end{array}$$

The indicator collector clauses are defined as in the local acceptability encoding, and are omitted here. In the special case that d is a power of 2, there is no need for indicator clauses or z_v, since there is no way to obtain unacceptable assignments of universal variables. In that case we omit the local acceptability variables entirely from the encoding. This is an unimportant change: if our encoding was used as presented without change, each z would be set false by unit propagation before search started.

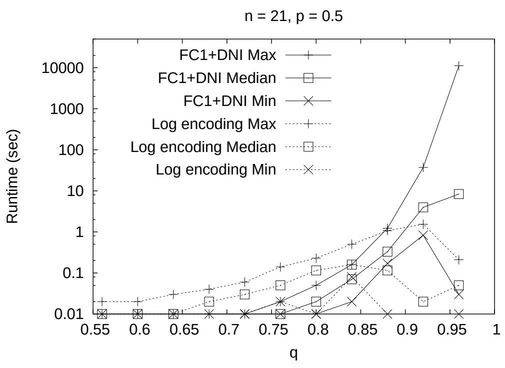

Figure 1. Comparison of the adapted log encoding with FC1+DNI

The subtlety of this encoding is that we omit the clauses which force equivalence between x_i^v and the corresponding values of b_j^v. So we omit clauses such as $z_u \vee \neg x_1^v \vee \neg b_2^v$. It might seem that this is erroneous, as it allows a universal to take two values, if x_1^v and x_2^v are both true. But there is no way that setting of the universal b_j^v can *force* more than one x_i^v to be true, On the other hand, consider the extreme case where truth is found with all x_i^v true, as for example can happen if v occurs in no constraints. This makes all log value clauses satisfied by x_i^v. The benefit of this is that the pure literal rule for existential variables can work. If x_i^v only occurs positively, it can be set to true. If this is the case for all values i of v, the b_i^v variables will disappear from the problem, greatly reducing the need for search. So, with sufficient care, we can use the pure literal rule included in our QBF solver, and have it work in the QCSP case. Apart from writing the translator, this is done without any coding effort on QCSPs.

The definition of acceptable assignment is revised to exclude inappropriate assignments to the b_i^v variables. After checking that unacceptable assignments have the properties required in Theorem 3, the rest of that proof can be used with almost no change. The only subtlety is the ability to set two x_i^v's true. As argued above, this can be done only when it makes no difference to the truth value of the problem. So we state without proof:

Theorem 4 *A QCSP is true if and only if the encoded QBF is true, for the adapted log encoding.*

Like local acceptability, the log encoding performs FC0 propagation.

The improvement the adapted log encodings gives over the previous encodings is extraordinary. It is now not only competitive with FC1+DNI (the best algorithm reported in [11]) but can be orders of magnitude better. Figure 1 shows results using the two methods, with the minimum, median and maximum of the 100 samples at each point. We see that there is a crossover at about $q = 0.86$. For smaller q, FC1+DNI is clearly better, although the worst case time of the QBF solving is never more than a second. For $q > 0.84$, the log encoding gives better results, and very dramatically so. At $q = 0.96$ the worst case time for the QBF encodings was 0.21 sec, compared to a median of 8.36 sec but a worst case of 11,173 sec for FC1+DNI.

It is obviously pleasing to be able to report more than four orders of magnitude improvement. And while QCSP techniques can be faster in Figure 1, this only happens on easy problems while the QBF technique is better on hard ones. More importantly, perhaps, we can use these results to suggest algorithmic improvements for QCSP solving. In particular, we suggest that the log encodings ability to exploit the pure literal rule in QBF may be important, and perhaps this can be incorporated into QCSP techniques.

6 Flaws in QCSPs

In our experiments at $n = 21$, we did not see the typical phase transition one might expect. The only data set at $p = 0.5$ with any soluble problems was the highest possible value of $q = 0.96$, and only 58% of instances were soluble. We now show that this is because the k-QCSP generator is subject to a flaw similar to those found in some generators of CSPs [1] or QBF [9]. Suppose we can find assignments $v_1 = 7, v_2 = 2, \ldots v_k = 3$, and an existential e quantified later than all the v_i. If every value of e conflicts with one of the chosen values of one of the universals, this assignment is false. But it remains false irrespective of assignments to other universals or existentials and so the problem is trivially false as a whole. Even taking the extreme case of only one conflict per constraint, this can happen as long as there are many universals before e as values in its domain. In this case, i.e. $q = 1 - 1/d^2$, if there is a constraint between variables with probability p, the probability of a flaw between e and d particular universals is $P(e) = \prod_{i=0}^{d-1} p(d-i)/d$. With k existential variables quantified inside d universals, the probability of no flaw occurring is $(1 - P(e))^k$. Since $P(e)$ does not depend on k, with fixed d and p this probability tends to 0 as $k \to \infty$. Not only are flaws certain to occur, but there is no phase transition: i.e. for any $p > 0$ almost all problems are false asymptotically.

The flaw dramatically affects experiments in the k-QCSP model at small problem sizes. At $n = 39$, $p = 0.5$, $q = 0.96$, each one of 100 random instances had the flaw. At $n = 21, p = 0.5, q = 0.96$, 42 problems were flawed: that means that all instances were either flawed or soluble. We conclude that there is an urgent need for an unflawed QCSP generation method in order to provide more representative test instances. In the absence of such a method, we have reported results with k-QCSP in order to compare the specialised QCSP algorithms with encodings into QBF. This comparison is still fair, as neither technique takes explicit advantage of the flaw. However, it is possible that translation into QBF allows advanced techniques such as conflict-directed backjumping to make flawed problems easy: if so, this highlights the advantages of translating new problems such as QCSP into mature domains like QBF.

Note that the flaw is simply a situation in which search can be terminated. As such it might give rise to interesting new propagation techniques in QCSP, or valuable new clauses in QBF encodings.

7 Discussion & Conclusions

We suggested that at an early stage into research into a new problem like QCSP, encoding into a more studied problem like QBF would provide competitive performance. We showed this to be the case, and indeed showed that on some instances we could do four orders of magnitude better than a specialised QCSP algorithm. However, these excellent results should not be oversold, as our identification of a flaw in the generator might mean that performance would be reversed on more representative instances.

We also suggested that performance of encodings and techniques used in QBF solving might indicate directions for future research into QCSP solving techniques. Our very good results on the log encoding suggest two areas to investigate more directly for QCSPs. First, we suggested that the success of the log encoding was its ability to take advantage of the pure literal rule in QBF, so introducing an analogue of that into QCSPs might be effective. Second, we used conflict-directed backjumping in our solver [12, 10], and this could also be incorporated into QCSP algorithms. On the other hand, we found that solution-directed backjumping [10] did not help search, so extending that technique to QCSP does not seem to be a priority.

What about other encodings of CSPs into SAT? The 'support encoding', in which clauses express support instead of conflicts, has been shown to be more effective than the direct encoding in SAT, because unit propagation in the encoded instance establishes arc consistency in the CSP [7]. The support encoding can easily be lifted to work in the global and local acceptability encodings. We need to include AMO clauses for existentials, and we replace the conflict clauses with support clauses, exactly as in SAT excepting the addition of indicator variables z. Like their direct counterparts, these encodings perform very badly. Unfortunately, we cannot simply replace conflict with support clauses in the adapted log encoding. This is because the encoding does not equate x_i^v with the relevant b's, giving the freedom of more than one x_i^v being true. If more than one are true, both can be used to support two existential variables, while neither on its own would support both. Thus correctness is lost. We can include clauses to force at most one x_i^v to be true, but then we lose the ability to set x_i^v by pure literal, which we found to be a key advantage in the direct version of the adapted log encoding. Informal experiments confirm that performance is extremely poor compared to the direct version of adapted log. Finding an effective analogue of the support encoding remains an open research problem.

ACKNOWLEDGEMENTS

We thank Nikos Mamoulis and Kostas Stergiou for the use of their instance generator and QCSP code. The first author is supported by a Royal Society of Edinburgh SEELLD Support Fellowship, the second by an EPSRC Doctoral Training Grant, and the third by EPSRC grant GR/55382. The authors are members of the APES and CP Pod research groups and thank other members, and our reviewers.

References

[1] D. Achlioptas, L.M. Kirousis, E. Kranakis, D. Krizanc, M.S.O. Molloy, and Y.C. Stamatiou, 'Random constraint satisfaction: A more accurate picture', in *Proc. CP97*, pp. 107–120. Springer, (1997).

[2] L. Bordeaux and E. Monfroy, 'Beyond NP: Arc-consistency for quantified constraints', in *Proc. CP-2002*, pp. 371–386, (2002).

[3] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi, 'An algorithm to evaluate quantified boolean formulae and its experimental evaluation', *Journal of Automated Reasoning*, **28**(2), 101–142, (2002).

[4] P. Jeavons F. Boerner, A. Bulatov and A. Krokhin, 'Quantified constraints: algorithms and complexity', in *Proc. CSL-2003*, (2003).

[5] A. Frisch and T.J. Peugniez, 'Solving non-boolean satisfiability problems with stochastic local search', in *Proc. IJCAI-01*, pp. 282–288, (2001).

[6] R. Genisson and P. Jegou, 'Davis and Putnam were already forward checking', in *Proc. ECAI-96*, pp. 180–184, (1996).

[7] I.P. Gent, 'Arc consistency in SAT', in *Proceedings of ECAI 2002*, pp. 121–125. IOS Press, (2002).

[8] I.P. Gent and A.G.D. Rowley, 'Encoding connect-4 using quantified boolean formulae', in *Modelling and Reformulating Constraint Satisfaction Problems*, ed., A.M. Frisch, pp. 78–93, (2003).

[9] I.P. Gent and T. Walsh, 'Beyond NP: the QSAT phase transition', in *Proceedings of the 16th National Conference on AI*, pp. 648–653. American Association for Artificial Intelligence, (1999).

[10] E. Guinchiglia, M. Narizzano, and A. Tacchella, 'Backjumping for quantified Boolean logic satisfiability', in *IJCAI-01*, pp. 275–281, (2001).

[11] Nikos Mamoulis and Kostas Stergiou, 'Algorithms for quantified constraint satisfaction problems', Technical Report APES-78-2004, APES Research Group, (January 2004).

[12] P. Prosser, 'Hybrid algorithms for the constraint satisfaction problem', *Computational Intelligence*, **9**, 268–299, (1993).

[13] T. Walsh, 'SAT v CSP', in *Proc. CP-2000*, pp. 441–456, (2000).