# Redundant modeling for the QuasiGroup Completion Problem

Iván Dotú, Alvaro del Val, and Manuel Cebrián

Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
`ivan.dotu@ii.uam.es, delval@ii.uam.es, meathook@terra.es`

**Abstract.** The Quasigroup Completion Problem (QCP) is a very challenging benchmark among combinatorial problems, and the focus of much recent interest in the area of constraint programming. [5] reports that QCPs of order 40 could not be solved by pure constraint programming approaches, but could sometimes be solved by hybrid approaches combining constraint programming with mixed integer programming techniques from operations research. In this paper, we show that the pure constraint satisfaction approach can solve many problems of order 45 in the transition phase, which corresponds to the peak of difficulty. Our solution combines a number of known ideas –the use of redundant modeling [3] with primal and dual models of the problem connected by channeling constraints [13]– with some novel aspects, as well as a new and very effective value ordering heuristic.

## 1 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. It has a broad range of practical applications [5]; it has been put forward as a benchmark which can bridge the gap between purely random instances and highly structured problems [6]; and its structure as a multiple permutation problem [13] 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.

In this paper, we present several techniques that together allow us to solve significantly larger QCPs than previously reported in the literature. Specifically, [5] reports that QCPs of order 40 could not be solved by pure constraint programming approaches, but could sometimes be solved by hybrid approaches combining constraint programming with mixed integer programming techniques from operations research. We show that the pure constraint satisfaction approach can solve many problems of order 45 in the transition phase, which corresponds to the peak of difficulty. Our solution builds upon some known ideas, such as the use of redundant modeling [3] with primal and dual models of the problem connected by channeling constraints [13], with some new twists. For example, we will consider models consisting of only channeling constraints, without any primal or dual constraints, and we demonstrate empirically for the first time the usefulness of channeling constraints linking several pairs of models of a problem, an idea that was considered, but only theoretically, in [15] and [14]. In addition, we present a new value ordering heuristic which proves extremely effective, and that could prove useful for many other problems with multiple models. The idea underlying this heuristic, which originates in the work of [15, 11] for single permutation problems, is that selecting a value for (say) a primal variable is in practice in the presence of channeling constraints also a choice of the dual variables corresponding to that value; therefore we can use *variable* selection heuristics on the dual variables to choose the *value* to assign to the previously chosen primal variable. Finally, we show how redundant constraints can be used to "compile arc consistency into forward checking", that is, to ensure that the latter has as much pruning power as the former but at a much lesser cost in constraint checks.

It is interesting to note that our approach involves only binary constraints, which seems to go against common wisdom about their limitations —when contrasted with the use of non-binary constraints such as alldiff [8]— in solving quasigroup completion problems [9]. It is certainly an interesting issue, which we plan to address in the future, whether the use of alldiff could yield even better results than our approach when coupled with other ideas in this paper.[1]

The idea of redundant modeling was first introduced by [3]. The benefits of adding redundant constraints to some given model to improve pruning power were well-known in the literature, but [3] went a step further by considering the redundant combination of full models of a problem, where the models may involve different sets of variables. This combination is achieved by specifying how the various models relate to each other through *channeling constraints*, which provide a mapping among assignments for the different models. The combined model contains the original but redundant models as submodels. The channeling constraints allow the sub-models to cooperate during constraint-solving by propagating constraints among the problems, providing an extra level of pruning and propagation which results in a significant improvement in performance.

Another important modeling idea that we use is that of permutation problems (see e.g. [11, 13]). A constraint satisfaction problem (CSP) is a permutation problem if it has the same number of variables as values, all variables have the same domain and each value can be assigned to a unique variable. Thus, any solution can be seen as assigning a permutation of the values to the variables. In the same manner, a multiple permutation problem has some (possibly overlapping) sets of variables, each of which is a permutation problem. QCP is a paradigmatic example of a multiple permutation problem.

The structure of this paper is as follows. We introduce in Section 2 the quasigroup completion problem (QCP) and present a number of alternative models

[1] Besides the obvious computational limitations in running large experimental suites of hard QCP problems, we were limited in this aspect by the unavailability of open source alldiff code.

that can be used to represent it; we then consider in Section 3 various ways of combining these models. Section 4 presents some experimental data to compare the relative merits of the various models, which will lead us to choose one particular model as the best one among those tested. Sections 5 and 6 are the core of the paper, where we apply the new value ordering heuristic to QCPs and then further tune our solution by adding some redundant constraints that allow us to replace the relatively expensive arc consistency with forward checking. Section 7 concludes the paper with some ideas for further research.

## 2 Models of quasigroups

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 $Q$ [5]. 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. Early work on quasigroups focused on quasigroup existence problems, namely the question whether there exist quasigroups with certain properties, solving several significant open mathematical problems [10]. We focus instead on the quasigroup completion problem (QCP), which is the (NP-complete [4]) problem 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 [6]. 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 [7].

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 [13]. 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).

QCPs appear in a number of real world applications such as conflict-free wavelength routing in wide band optical networks, statistical design, and error correcting codes [5].

### 2.1 Models

Let $P$ be a problem. To model $P$ as a CSP we need to fix a set of variables $X$, a function $F$ that maps each variable $x_i \in X$ to a domain of possible values, and a set of constraints $C$ defined over the variables in $X$, so that the set of solutions in the traditional CSP sense corresponds in some exact mathematical sense to the solutions of $P$, if any. The triple $(X, F, C)$ is then a *model* of $P$. There is usually more than one model for any given problem, but whichever we choose, we need to ensure that it fully characterizes the problem. Cheng et al. [3] define two models $M_1 = (X_1, F_1, C_1)$ and $M_2 = (X_2, F_2, C_2)$ of a problem $P$ to be *redundant* when the following conditions hold:

1. $M_1$and $M_2$ are models of $P$ respectively, i.e. each of them fully characterizes the set of solutions to $P$.
2. $X_1 \cap X_2 = \emptyset$.

Redundancy is a double-edged sword: it can help propagation by allowing more values to be pruned at any given point in the search, but it can also hinder it by forcing it to process a larger set of constraints. Fortunately, more fine grained distinctions are possible, as we might choose to combine only parts of various models. We could not speak of combining models if we don't use their respective sets of variables, but it will often be advantageous (as we will see) to drop some of the constraints from one or more models that become redundant when making the combination. If we do this, however, we must be careful to ensure the correctness and completeness of the combined model.

Several models can be defined for QCPs, as described next. While all models have the same logical status, it is common to distinguish between *primal* and *dual* models. The distinction is only a matter of perspective, specially in permutation problems, where variables and values are completely interchangeable.

## 2.2 Primal Model

The primal model for QCP, as usually defined, takes variables to represent the cells of the Latin Square for a QuasiGroup, and the domains of possible values consist of the colors to be assigned. Thus, the *primal variables* are the set $X = \{x_{ij} \mid 1 \leq i \leq n,\ 1 \leq j \leq n\}$ where $x_{ij}$is the 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. All variables share a common initial domain, namely $D = \{k \mid 1 \leq k \leq n\}$, where each $k$ represents a color. The *primal constraints* in turn can be divided into row constraints and column constraints. If we choose a binary representation, there are $n^2$ row constraints of the form $x_{ij} \neq x_{il}$ where $x_{ij}, x_{il} \in X$ and $j \neq l$, which means that two cells in the same row must not have the same color; and $n^2$ column constraints of the form $x_{ij} \neq x_{lj}$ where $x_{ij}, x_{lj} \in X$ and $i \neq l$, which means that two cells in the same column must not have the same color. Equivalently, we could use just $2n$ *alldiff* constraints [8], one for each row and column. Semantically this makes no difference.

The primal model (or `pr` model for short) provides a complete characterization of the problem.

## 2.3 Row Dual Model

There are different ways to formulate dual models for a multiple permutation problem. Here we consider dual models for each of the permutation subproblems (as opposed to a single dual model of the primal problem), and group them by row and column, to obtain two complete models of QCPs. In the *row dual model*, the problem is reformulated as the question of which position (column) in a given row has a given color. The *row dual variables* are the set $R = \{r_{ik} \mid 1 \leq i \leq n,\ 1 \leq k \leq n\}$ where $r_{ik}$ is the $k$th color in the $i$th row. The domain of each variable is again the set $D = \{j \mid 1 \leq j \leq n\}$, but now the values represent columns, i.e. the positions in row $i$ where color $k$ can be placed. The *row dual constraints* are similar to the primal constraints. There are $n^2$ constraints of the form $r_{ik} \neq r_{il}$, where $r_{ik}, r_{il} \in R$ and $l \neq k$, which means that two colors in the same row must not be assigned to the same column; and $n^2$ constraints of the form $r_{ik} \neq r_{jk}$ where $r_{ik}, r_{jk} \in R$ and $i \neq j$, which means that the same color in different rows must not be assigned to the same column. Alternatively, we could have *alldiff*$(r_{i1}, \ldots, r_{in})$ for every row $i$, and *alldiff*$(r_{1k}, \ldots, r_{nk})$ for every color $k$.

A simple symmetry argument shows that this model also fully characterizes the problem.

### 2.4 Column Dual Model

The second dual model is composed of the set of dual models for each column permutation constraint, representing the colors in each column. The *column dual variables* are the set $C = \{c_{jk} \mid 1 \leq j \leq n,\ 0 \leq k \leq n\}$ where $c_{jk}$ is the $k$th color in the $j$th column. All variables have domain $D = \{i \mid 1 \leq k \leq n\}$, where $i$ represents the rows where color $k$ can be placed in the $j$th column. Similar to the row dual model, we have *column dual constraints* of the form $c_{jk} \neq c_{jl}$ where $c_{jk}, c_{jl} \in C$ and $k \neq l$, which means that two colors in the same column must not be assigned to the same row; and of the form $c_{jk} \neq c_{lk}$ where $c_{jk}, c_{lk} \in C$ and $j \neq l$, which means that the same color in different columns must not be assigned to the same row.

This model also fully characterizes the problem. We refer to the combination of both dual models as the `dl` model.

## 3 Combining the Models

A *channeling constraint* for two models $M_1 = (X_1, F_1, C_1)$ and $M_2 = (X_2, F_2, C_2)$ is a constraint relating variables of $X_1$ and $X_2$ [3]. We will consider the following kinds of channeling constraint:

- *Row Channeling Constraints:* Constraints for the $n$ row permutation constraints, linking the primal model with the row dual model:
$$x_{ij} = k \Leftrightarrow r_{ik} = j.$$
- *Column Channeling Constraints:* Corresponding to the $n$ column permutation constraints, they link the primal and the dual column models:
$$x_{ij} = k \Leftrightarrow c_{jk} = i.$$

– *Triangular Channeling Constraints:* These constraints link both dual models, closing a "triangle" among the three models:

$$c_{jk} = i \Leftrightarrow r_{ik} = j.$$

Given two or more redundant, complete models, we can obtain a combined model by simply implementing all the models and linking them by channeling constraints. Thus the full combined model or `pr-dl-ch2-model` resulting from the above models is the model consisting of primal and dual variables and constraints, linked together by row and column channeling constraints.[2] More generally, as long as a combined model includes a complete model of the problem as a submodel, we are free to add any set of variables or constraints from other models, with the only requirement that in order to add a constraint all its variables must belong to the combined model. Thus, for example, given the primal variables and constraints, we may choose to add any number of dual and channeling constraints as long as the corresponding variables are also added. For example, we may decide to use only the row dual variables together with the row dual constraints and/or row channeling constraints. Nothing is lost by not including parts of the dual models, since all the necessary information is present in the primal model.

In fact we can take this as far as removing all primal and dual constraints! Walsh [13] shows that arc consistency on the channeling constraints for a permutation problem dominates in pruning power over arc consistency over the binary not-equal constraints. Intuitively, this means that nothing is gained by adding the not-equal constraints once we have the channeling constraints. Note that this doesn't prove the superiority of a model with only channeling constraints over, say, the primal model, as the former also has many more variables and constraints; this issue is empirically examined later. It is important however to show that the model consisting of primal and dual variables, with *only* row and column channeling constraints, but *without* the primal or dual constraints (i.e. alldiff or not-equal) is also a complete model of the problem. We refer to this model as the *bichanneling model* or `ch2`:

**Proposition 1.** *The bichanneling model is equivalent to the primal model, hence it provides a full characterization of QCPs.*

*Proof.* If the two models had the same set of variables and associated domains, we could define equivalence just as having the same set of solutions. Since that's not the case here, we need to provide instead a one-to-one mapping between solutions of either model.

Let us say that a primal assignment, or P-assignment for short, is an assignment of values to all the primal variables, and a PD-assignment an assignment to all primal and dual variables.

The proposition can then be phrased more exactly in terms of the following two claims.

[2] We don't consider adding the triangular constraints until later.

Claim 1: Any P-assignment A which satisfies the (primal) alldiff constraints can be extended to a PD-assignment B which satisfies the channeling constraints. To extend A to B, we just pick each label $x_{ij} = k$ from A and set $r_{ik} = j$ and $c_{jk} = i$ in B. To see that B is well-defined, note that every $r_{ik}$ gets assigned, since A must use all available colors in order to fill row $i$ in accordance with the primal constraints; and that any given $r_{ik}$ is assigned at most once, since otherwise we would have $x_{ij} = x_{ih}$ for distinct columns $j$ and $h$, in contradiction with the fact that A satisfies the primal constraints. Similarly for any $c_{jk}$. Hence B is well-defined, and it satisfies the channeling constraints by construction.

Claim 2: Any PD-assignment B satisfying the row and column channeling constraints, is such that its primal subset A satisfies the primal constraints. Suppose not. Then B assigns the same value $k$ to two primal variables $x_{ij}$ and $x_{ih}$ for $j \neq h$ (or the completely symmetric case where it is row indexes that vary). But since B satisfies the row channeling constraints, $B$ should satisfy $r_{ik} = j$ and $r_{ik} = h$, which is impossible. $\square$

Yet another combined model we will consider later is the *trichanneling model*, or `ch3` for short, which adds the triangular channeling constraints to `ch2`, but still keeps away from the primal and dual constraints. Given the above proposition, `ch3` is also a complete model, and redundantly so.

## 4 Comparing models

Our initial results on the various models were in fact quite favorable to the bichanneling model. In order to present them, we need to say a few words about the experiments in this paper. First, in order to make our results comparable with others appearing in the literature, all instances were generated using the *lsencode* generator of QCPs, kindly provided to us by Carla Gomes. This generator begins by randomly coloring an empty quasigroup using a local search algorithm, and then randomly decoloring some cells. Hence all problems in our suites have a solution. All instances are of the "balanced" kind, which are known to be the hardest [5]; and most instances correspond to problems with 60% cells preassigned, which is close to the transition phase and corresponds to a peak in problem hardness. Second, all experiments are run with a slightly optimized variant of van Beek's GAC library, which comes as part of the CSP planning system CPLAN [12], and which implements generalized arc consistency (though in our case we only need its binary version, i.e. MAC [1]). As discussed below, neither CBJ nor nogood learning seem to help in QCP, contrary to the experience in many other domains, hence they are disabled in our tests. Also, all experiments use the min-domain variable selection heuristic, which we found to be uniformly the best among the ones we tried (see also [3, 11] and the discussion in Section 5).

In our initial tests, we found that the bichanneling model `ch2` could solve many problems that were out of reach for the other models, including many order 35 and some order 40 quasigroups with 60% preassigned cells. Table 1 shows mean time for solved instances and median time for the whole sample, both in seconds, and percent of solved instances within the given timeout (also in seconds) for sets of 50 instances of orders 30, 35 and 40, and 20, 42, 60 and 80% preassignment. (These results are also plotted in Figure 1 later.)

| % preassign → | | 20% | | 42% | | 80% | |
|---|---|---|---|---|---|---|---|
| order | % solved | mean | median | mean | median | mean | median |
| 30 | 100% | 0.94 | 0.93 | 0.43 | 0.25 | 0.03 | 0.02 |
| 35 | 100% | 1.99 | 1.99 | 0.71 | 0.53 | 0.05 | 0.05 |
| 40 | 100% | 4.98 | 4.98 | 2.51 | 1.09 | 0.08 | 0.08 |

| | 60% preassigned | | | |
|---|---|---|---|---|
| order | % solved | timeout | mean (solved) | median (all) |
| 30 | 18% | 100 | 48.74 | 100 |
| 35 | 22% | 3600 | 903.07 | 3600 |
| 40 | 10% | 3600 | 1751.90 | 3600 |

**Table 1.** Experimental results for the bichanneling model, MAC, no value ordering.

Our data confirm the existence of a peak of difficulty around 60% preassignment [5], whereas problems were trivially solvable with all other percentages we tried. Even though the results were promising, specially when compared with other models, they were also disappointing, in that the number of problems that we could solve in the transition phase was rather limited for various dimensions. (Note that in these cases, median time is the same as timeout because less than 50% of instances were solved.) Nevertheless, we decided to pursue further the bichanneling model based on the somewhat anecdotal evidence of its clear superiority over other models. As the following sections show, we succeeded in this goal.

For the sake of a more systematic comparison, we present here a simple comparison of the various models. Due to limited available time, we chose the 29 easiest problems (as measured with the approaches developed later) for order 30 quasigroups with 60% preassignment. These are still relatively difficult problems in the phase transition: the `ch2-model` took a total of 6624 seconds on the 19 problems (66%) in the sample that were solved with all tested models in less than 1800 seconds, yielding an average of 348.6 seconds per solved problem, and a mean (over the whole sample) of 574.16s. Table 2 shows the result of a comparison between various models on this sample. The table provides the ratios in the accumulated data in time and constraint checks over the solved problems, relative to the performance of `ch2`. Note that all models tried exactly the same number of assignments in all problems, empirically confirming the fact that arc consistency has identical pruning power in all four models.

We conjecture that these ratios will increase with problem difficulty. But there is little point on belaboring these data, as much better solutions are available, as discussed in the following sections.

| pr | | pr-dl | | pr-dl-ch2 | | ch2 | |
|---|---|---|---|---|---|---|---|
| time | checks | time | checks | time | checks | time | checks |
| 1.45 | 1.30 | 1.93 | 1.69 | 1.90 | 1.69 | 1 | 1 |

**Table 2.** Comparison of various models using MAC and no value ordering.

## 5 Variable and Value Ordering

It is well know that the order in which we make our choices as to which variable to instantiate, and with which value, can have a major impact in the efficiency of search. As already pointed out, all the results reported in this paper use the *min-domain* variable ordering heuristic (often denoted `dom`), which at each search node chooses a variable with the smallest domain to instantiate. The reason for this is simply that we obtained better results with it than with other alternatives we tried. These included more fine-grained heuristics such as `dom+degree` and `dom/degree`, yielding further confirmation to previous results by [3] and [11] on simple permutation problems. These other heuristics would often make no difference with respect to `dom`,[3] but when they did it was most often to the worse. (We did not perform a systematic comparison, though.) We also considered a number of variants of the above which took into account the (primal or dual) model to which variables belong, e.g. selecting only among primal variables, or only among primal variables unless some dual variable had a singleton domain, etc. These variants would often significantly underperform the previous ones, so we didn't pursue them further.

[15] introduced a *min-domain value ordering heuristic* for use when dual variables are available during the search. The idea is to choose the value such that the corresponding dual variable has the smallest current domain. To generalize this idea to multiple permutation problems, we need a way to take into account the two dual models. The one that worked best is what we might call the *min-domain-sum value selection heuristic* (or more briefly `vdom+`, the 'v' standing for value). Once a primal or dual variable is selected, we need to choose a value for it. Since any such value corresponds to one specific variable from each of the two other models, we select the value whose corresponding two variables have a minimal "combined" domain. Specifically, say we have chosen $x_{ij}$. Then we choose a color $k$ from its currently active domain for which the sum of the current domain sizes of $r_{ik}$ and $c_{jk}$ is minimal among the currently available colors for $x_{ij}$. Similarly, if the chosen variable is a dual one, say $r_{ik}$, we choose a column $j$ for this variable as a function of the current domain sizes of the corresponding variables $x_{ij}$ and $c_{jk}$.

[3] This is not much of a surprise, since the degree of a variable (number of constraints in which it is initially involved) cannot discriminate much among variables in a QCP; though this could also depend on details of implementation such as whether constraints are generated for variables that are explicitly or implicitly assigned by the initial coloring.

The results when the first combined model was used with the min-domain-sum value ordering heuristic were quite surprising, as it outperformed previous tests in three orders of magnitude in some cases. For example, for the instance bqwh-35-405-5.pls (balanced instance of order 35 and 60% preassigned cells) it took 2905 secs without value ordering and only 0.40 secs with it. For a more general picture, Figure 1 plots the data of Table 1, obtained with lexicographic value ordering, against the results over the same sample with `dom+` value ordering.

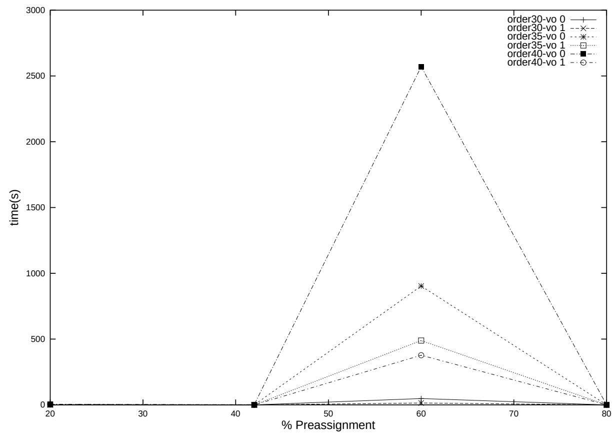


**Fig. 1.** Mean solution time on QCPs of order 30, 35 and 40 with (vo1) and without (vo0) value ordering.

Encouraged by this performance, we generated a set of 100 balanced instances of orders 30, 35, 40 and 45, with 60% preassignment. Table 3 shows median and mean time in seconds (the latter taken only over solved instances), percent of solved instances and timeouts, in solving these instances with the new variable ordering heuristic.

| order | mean | median | % solved | timeout |
|---|---|---|---|---|
| 30 | 148.84 | 174.11 | 68% | 1000 |
| 35 | 533.43 | 163.48 | 84% | 3600 |
| 40 | 732.94 | 1010.82 | 68% | 5000 |
| 45 | 1170.81 | 2971.40 | 56% | 6000 |

**Table 3.** The min-domain value ordering heuristics at the phase transition, using MAC.

These results are significantly better than those previously found in the literature, as we can solve over 50% of balanced QCPs of order 45 at the phase transition. Recall that, as pointed out in the introduction, [5], reports that pure constraint programming approaches, even when using specialized forms of arc consistency for non-binary alldiff constraints and a commercial solver, could not solve any problem of order 40 in the phase transition.

We considered other ways of combining domain sizes such as minimizing the product of the corresponding domain sizes (*min-domain-product* or `vdom*`), and their corresponding maximizing versions, without success. Perhaps there is no deep reason why `vdom+` was so clearly superior to `vdom*`. Maximizing versions were clear underperformers, and there is a reasonable explanation for it. For concreteness, consider choosing a value with the maximal combined domain of the corresponding variables, e.g. a value $k$ for a primal variable $x_{ij}$ such that $\textit{domain-size}(r_{ik}) + \textit{domain-size}(c_{jk})$ is maximal (over the colors available for $x_{ij}$ at the current stage of search). While large domain sizes are usually indication of less tightness, and thus could be conjectured to capture the idea, often cited in connection with value ordering, of selecting a value which is "more likely to lead to a solution", in this case they have exactly the opposite effect. When $x_{ij} = k$ is the maximal labeling according to this criteria, the domains of $r_{ik}$ and $c_{jk}$ are immediately pruned into singletons. Hence a maximizing choice produces maximal pruning, which is the opposite of what is desired. And conversely, heuristics such as `vdom+` choose values that produce the least pruning.

## 6 Compiling AC to FC with redundant constraints

Our next and last step in improving our solution derived from an examination of the pruning behavior of the bichanneling model with arc consistency. Suppose $x_{ij}$ is assigned $k$ at some point during the search. The GAC implementation of CPlan begins by checking arc consistency for constraints with a single uninstantiated variable, i.e. doing forward checking, which forces the domains of $r_{ik}$ and $c_{jk}$ to become the singletons $\{j\}$ and $\{i\}$ respectively, and also prunes, for each $h \neq k$, $j$ from $r_{ih}$, and $i$ from $c_{jh}$. Arc consistency will further discover (if not already known at this stage of the search):

- $x_{ih} \neq k$ for any column $h \neq j$, since otherwise $r_{ik} = h \neq j$;
- hence also $c_{hk} \neq i$ for any column $h \neq j$, since otherwise $x_{ih} = k$;
- similarly, $x_{hj} \neq k$ for any row $h \neq i$, since otherwise $r_{ik} = h \neq j$;
- hence also $r_{hk} \neq j$ for any row $h \neq i$, since otherwise $x_{hj} = k$;

It is not difficult to show that GAC cannot prune any more values as a result of an assignment to a primal variable, unless one of the listed prunings reduces a domain to a singleton. All these are useful prunings, but GAC does much more work than needed to obtain them. Each one of the pruned values – one for each $x_{ih}, x_{hj}, c_{hk}, r_{hk}$, potentially $4(n-1)$ pruned values and variables from a single assignment – requires GAC to check all the constraints in which the corresponding variables are involved, namely $2(n-1)$ or $(n-1)$ constraints for, respectively, the primal and dual pruned variables (further, in the CPlan implementation all affected variables have *all* their values tested, even if at most one will be pruned). This is wasted effort, as no additional pruning is achieved. One can however observe that most of the pruning power can be derived simply by assigning the variables whose domain became singletons (either directly through channeling constraints or indirectly when pruning a single value results in a singleton) and doing forward checking on them. To see that the remaining values pruned by GAC (namely the second and fourth items above) are also pruned by FC with the trichanneling model, observe that $c_{hk} \neq i$ since otherwise $r_{ik} = h \neq j$ using the corresponding triangular channeling constraint, and similarly $r_{hk} \neq j$ since otherwise $c_{jk} = h \neq i$.

We remark that the same effect can be achieved in different ways, e.g. the bichanneling model supplemented with the dual not-equal constraints also allows forward checking to derive the same consequences.

Table 4 compares the bichanneling model `ch2`, using only row and column channeling constraints with GAC, versus the trichanneling model `ch3` with the three kinds of channeling constraints using only FC, in both cases with the min-domain-sum value ordering. Each sample consists again of 100 balanced instances with 60% preassignment;the accumulated values are over the problems solved by both approaches within the given timeout. The median times are on the other hand over the whole sample. Accumulated times are in seconds while the other accumulated values are in millions of checks and tried assignments respectively.

| | ch3-fc | | | ch2-ac | | | ratios | |
|---|---|---|---|---|---|---|---|---|
| order | acc. time | median | solved | acc. time | median | solved | acc. time | median |
| 30 | 6445.44 | 153.04 | 78% | 9557.83 | 174.11 | 68% | 1.48 | 1.14 |
| 35 | 29691.18 | 152.16 | 86% | 45341.22 | 163.48 | 85% | 1.53 | 1.07 |
| 40 | 33015.14 | 637.18 | 73% | 48682.04 | 1010.82 | 68% | 1.47 | 1.59 |
| 45 | 38569.95 | 1650.52 | 59% | 61469.78 | 2971.40 | 56% | 1.59 | 1.80 |

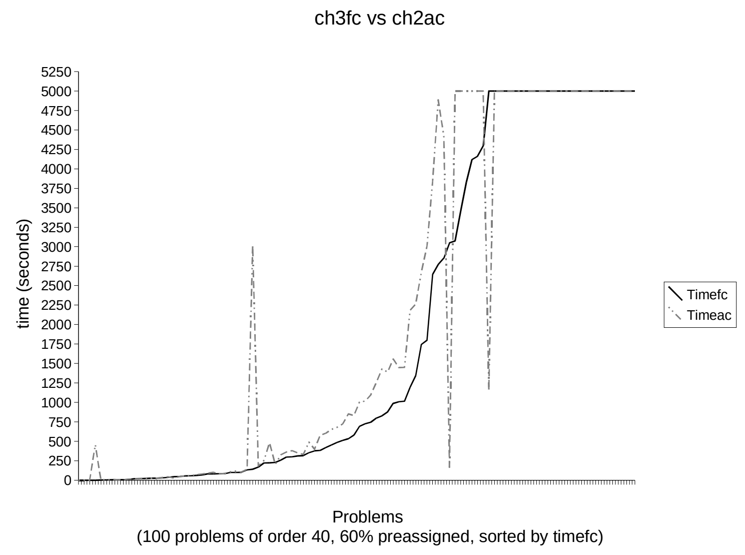
| | checks | | | visits | | |
|---|---|---|---|---|---|---|
| order | ch3-fc | ch2-ac | ratio | ch3-fc | ch2-ac | ratio |
| 30 | 29886 | 80206 | 2.68 | 431 | 658 | 0.15 |
| 35 | 114572 | 279003 | 2.44 | 1617 | 218 | 0.13 |
| 40 | 205247 | 445790 | 2.17 | 2769 | 331 | 0.12 |
| 45 | 108276 | 321632 | 2.97 | 1489 | 236 | 0.16 |

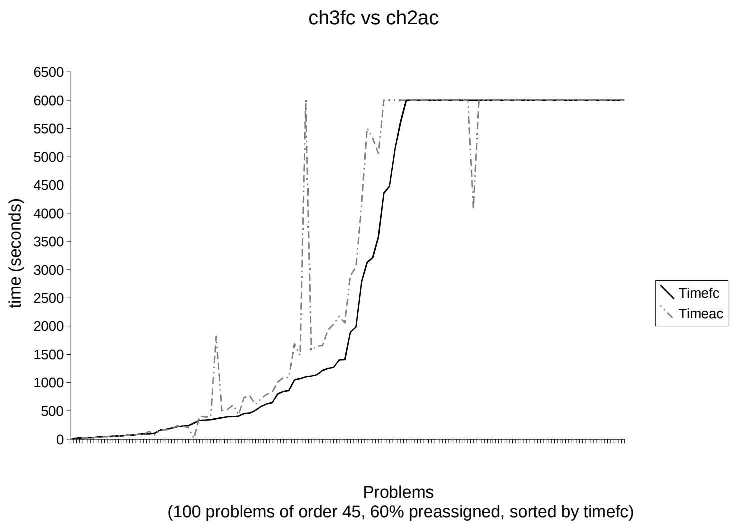
**Table 4.** The ch3 and ch2 models compared, with value ordering.

These tables show that there is a significant improvement in time with the `ch3` model using only FC, and this can be traced to the large savings in number of checks. On the other hand, `ch3` with FC tries almost one order of magnitude more assignments, which arise from the fact that it must instantiate the variables associated to a given assignment made in the search tree in order to extract the same consequences as AC with ch2; these added tried assignments do not however translate into any more checks or more true backtracking.

The results in this table are not however as straightforward to obtain as the formal result on the equivalent pruning power may suggest. Indeed, our first attempt at implementing ch3 resulted in a slight but noticeable slowdown! On further examination, we realized that this was due to the implementation of the min-domain variable ordering heuristic, which could select many other variables with a singleton domain before the variables associated with the last assignment; as a result, obtaining the same conclusions as AC could be significantly delayed. We solved the problem by keeping a stack of uninstantiated variables with singleton domain, and modifying the min-domain heuristic to pop the most recent variable from that stack whenever it was not empty. This ensures that FC considers those variables that have just become singletons immediately. The solution has nevertheless an ad-hoc flavor, and suggests that for domains such as QCPs, where propagation often forces a value for variables as opposed to merely pruning part of their domain, a more SAT-like propagation may be more indicated; in other words, it is not always sufficient to rely on the min-domain heuristic to propagate in a timely fashion forced values.

Finally, the following figures display a more detailed picture of how ch2 and ch3 compare, showing the time taken to solve all 100 problems in each set, sorted by difficulty, for order 40 and 45 quasigroups at the phase transition. As it can be seen, the ch3 model is almost always superior, but there are some anomalies that are worth investigating further.





## 7 Conclusions and future work

In summary, we have shown in this paper that a pure CSP approach can handle quasi-group completion problems significantly larger than was thought possible, using appropriate models, value ordering heuristics, and algorithms, even in the absence of global alldiff constraints. Our solution is arguably much simpler than the hybrid CSP/OR approach developed in [5] yet it seems to clearly outperform it, saving distances for different machines, implementations and execution environments. It would be interesting to see whether the combination of the ideas of this paper with either alldiff constraints or OR techniques could yield further improvements in our ability to solve larger QCPs. For example, we mentioned that the same effect achieved by introducing triangular channeling constraints would be achieved by reintroducing instead the dual not-equal constraints, which in turn could be replaced by dual alldiff constraints.

We have introduced two novel aspects within redundant modeling in multiple permutation problems:

– A novel value ordering heuristic which takes into account the primal and both dual models, and which generalizes for multiple permutation problems ideas introduced in ([15, 11] for simple permutation problems. The speedup produced by this heuristic is quite remarkable, up to three orders of magnitude in some cases.
– The use of channeling constraints linking more than a single pair of models to provide forward checking with the same pruning power as arc consistency

at a much smaller cost in constraint checks, and thus in performance, provided that ordering effects are taken into account in the min-domain variable selection heuristic.

Many issues remain to be explored. While we did try a number of alternatives to the presented value ordering heuristics without success, others may be more successful. There are some anomalies in the behavior of the ch3-fc approach vs ch2-ac which could be symptoms of more subtle effects than the ordering effects reported above, and which need to be explored. There is finally the issue of why CBJ and nogood learning did not help in this problem, which may in part suggest that in a sense randomness dominates over structure in QCPs, but which should at any rate be an incentive to develop more effective implementations of these techniques so that at least they do not hurt when they do not help.

## References

1. C.Bessiere and J.C.Regin. Mac and combined heuristics: two reason to forsake FC (and CBJ?) on hard problems. In *2nd Int. Conf. on Principles and Practice of Constraint Programming* pp. 61–75, 1996.
2. J.Bitner and E.M.Reingold Backtrack programming techniques. In *Communications of the ACM* 18: 651–655.
3. B.M.W. Cheng, J.H.M. Lee, and J.C.K Wu. Speeding up constraint propagation by redundant modeling. In *2nd Int. Conf. on Principles and Practice of Constraint Programming*, pp. 91–103, 1996.
4. C. Colbourn. The complexity of completing partial latin squares. In *Discrete applied Mathematics.*
5. C. Gomes and D. B. Shmoys. The promise of LP to Boost CSP Techniques for Combinatorial Problems. In *CP-AI-OR'02*, pp. 291–305, 2002.
6. C. Gomes and D. B. Shmoys. Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem In *Proc. Computational Symposium on Graph Coloring and Extensions*, 2002.
7. C. Gomes, B. Selman, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. In *Journal of Automated Reasoning*, 24:67-100, 2000.
8. J-C. Regin. A filtering algorithm for constraints of difference in CSPs. In *Proc. AAAI'94*, pp. 362–367, 1994.
9. K. Sergiou and T. Walsh. The difference all-difference makes. In *Proc. IJCAI'99.*
10. J. Slaney, M. Fujita and M. Stickel. Automated reasoning and exhaustive search: Quasigroup Existence Problems. In *Computers and Mathematics with Applications*, 29:115–132, 1995.
11. Barbara M. Smith. Modeling a Permutation Problem. In *Proceedings of ECAI'2000 Workshop on Modeling and Solving Problems with Constraints*, 2000.
12. P. van Beek and X. Chen. CPlan: A Constraint Programming Approach to Planning. In *AAAI'99*, pp. 585–590, 1999.
13. T. Walsh. Permutation Problems and Channeling Constraints. In *LPAR-2001.*
14. Barbara M. Smith. Dual Models in Constraint Programming. School Computing Research Report 2001.02, University of Leeds, January 2001.
15. B.M.W. Cheng, K.M.F. Choi, J.H.M. Lee, and J.C.K Wu. Increasing Constraint Propagation by Redundant Modeling: an Experience Report. In *Constraints*, pp. 167–192, Kluwer Academic Publishers,1999.