

Propagation Redundancy for Permutation Channels *

C.W. Choi and J.H.M. Lee

Dept. of Comp. Sci. & Eng.

The Chinese University of Hong Kong

Shatin, N.T., Hong Kong SAR, China

{cwchoi, jlee}@cse.cuhk.edu.hk

P. J. Stuckey

Dept. of Comp. Sci. & Soft. Eng.

University of Melbourne, 3010

Australia

pjs@cs.mu.oz.au

1 Introduction

Finding a good model of a constraint satisfaction problem (CSP) is a challenging task. A modeller must specify a set of constraints that capture the definitions of the problem, and the model should also have strong propagation. In other words, the model should be able to quickly reduce the domains of the variables of the problem, and the implementation of these propagators should be efficient, and the search space should not be too large.

A problem can be modelled differently from two viewpoints using two different sets of variables. In redundant modelling [Cheng *et al.*, 1999], we connect the two different models with channelling constraints, which relates valuations in the two different models stronger propagation behaviour can be observed. However, the additional variables and constraints impose extra computation overhead may outweigh the gain of reduction in search space.

In this paper we consider redundant models connected by permutation channels, which commonly arise when the underlying problem is some form of assignment problem. Since each model is complete and only admits the solutions of the problem, each model is logically redundant with respect to the other model plus the permutation channel. In order to keep the benefits of redundant modelling without paying all the costs, We give a theorem which allows us to determine when we can eliminate constraints in the mutually redundant models that do not give extra propagation. Due to space limitations, we state the theorem without proof.

2 Reasoning about Domain Propagation

We consider integer constraint solving with constraint propagation and tree search.

An *integer valuation* θ is a mapping of variables to integer values, written $\{x_1 \mapsto d_1, \dots, x_n \mapsto d_n\}$. Let *vars* be the function that returns the set of variables appearing in a constraint or valuation. A *constraint* c defines a set of valuations $\text{solns}(c)$ each mapping the same set of variables $\text{vars}(c)$. We

call $\text{solns}(c)$ the *solutions* of c . A constraint c is *logically redundant* with respect to a set of constraints C if $\models C \rightarrow c$.

A *domain* D is a complete mapping from a fixed (countable) set of variables V to finite sets of integers. A *false domain* D is a domain with $D(x) = \emptyset$ for some x . A domain D_1 is *stronger* than a domain D_2 , written $D_1 \sqsubseteq D_2$, if D_1 is a false domain or $D_1(x) \subseteq D_2(x)$ for all variables x . The initial domain D_{init} gives the initial values possible for each variable, allows us to restrict attention to domains D such that $D \sqsubseteq D_{\text{init}}$.

We adopt the notion of *propagation solver* and *domain consistency*¹ from Schulte and Stuckey [2001]. A *propagator* f is a monotonically decreasing function from domains to domains. A *propagation solver* for a set of propagators F and current domain D , $\text{soln}(F, D)$, repeatedly applies all the propagators in F starting from domain D until there is no further change in resulting domain. A domain D is *domain consistent* if D is the least domain containing all solutions of c in D . Define the *domain consistency propagator* $\text{dom}(c)$ for a constraint c such that $\text{soln}(\text{dom}(c), D)$ is always domain consistent for c .

For all domains $D \sqsubseteq D_{\text{init}}$, a set of propagator F_2 is made *propagation redundant* by a set of propagators F_1 , written $F_1 \gg F_2$, if $\text{soln}(F_1, D) \sqsubseteq \text{soln}(F_2, D)$, and is *equivalent* to F_1 , written $F_1 \approx F_2$, if $\text{soln}(F_1, D) = \text{soln}(F_2, D)$.

It is well known that in general the domain propagation of a conjunction of constraints is not equivalent to applying the domain propagators individually. But there are cases where propagation of a conjunction is equivalent to propagation on the individual conjuncts.

Lemma 1 *If c_1 and c_2 share at most one variable x , then $\{\text{dom}(c_1), \text{dom}(c_2)\} \approx \{\text{dom}(c_1 \wedge c_2)\}$.*

An *atomic constraint* is one of $x_i = d$ or $x_i \neq d$ where $x_i \in V$ and d is an integer. An atomic constraint represents the basic changes in domain that occur during propagation.

A *propagation rule* is of the form $C \rightarrow c$ where C is a conjunction of atomic constraints, c is an atomic constraint and $\not\models C \rightarrow c$. Note our propagation rules are similar to the “membership rules” of Apt and Monfroy [2001] except we allow equations on the right hand side.

A propagator f *implements* a propagation rule $C \rightarrow c$ if for each $D \sqsubseteq D_{\text{init}}$ whenever $\models D \rightarrow C$, then $\models f(D) \rightarrow c$.

¹Equivalently, hyper-arc or generalized arc consistent.

*We thank Barbara Smith and the anonymous referees for their constructive comments. The work described in this paper was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK4204/01E).

We can characterize a propagator f in terms of the propagation rules that it implements. Let $rules(f)$ be the set of rules implemented by f . Then $prop(f) \subseteq rules(f)$ are a set of propagation rules such that every $r \in rules(f)$ is subsumed by a rule $r' \in prop(f)$.

3 Permutation Channels

A common form of redundant modelling is when we consider two viewpoints to a permutation problem. We can view the problem as finding a bipartite matching between two sets of objects of the same size. For notational convenience, let the two viewpoints as having the set of variables $X = \{x_0, \dots, x_n\}$, and $Y = \{y_0, \dots, y_n\}$ respectively.

The *permutation channel* $C_{\bowtie}$ is defined by the conjunction of constraints $\bigwedge_{i=0}^n \bigwedge_{j=0}^n (x_i = j \Leftrightarrow y_j = i)$. The *permutation channel propagator* $F_{\bowtie}$ maintains domain consistency of each individual bi-implication, that is $\bigcup_{i=0}^n \bigcup_{j=0}^n \{dom(x_i = j \Leftrightarrow y_j = i)\}$.

Smith [2000] first observes that the permutation channel makes each of the disequations between variables in either model propagation redundant. Walsh [2001] proves this holds for other notions of consistency.

Lemma 2 (Walsh, 2001) $F_{\bowtie} \gg \{dom(x_i \neq x_k)\}$

Related to $C_{\bowtie}$ is the *permutation channel function* $\bowtie$ which is a bijection between atomic constraints in X to atomic constraints in Y , $\bowtie(x_i = j) = (y_j = i)$, and $\bowtie(x_i \neq j) = (y_j \neq i)$. We extend $\bowtie$ to map conjunctions of constraints in the obvious manner $\bowtie(C_1 \wedge C_2) = \bowtie(C_1) \wedge \bowtie(C_2)$.

The fundamental theorem states that a constraint in Y is propagation redundant if there exist a constraint in X when conjuncts with $C_{\bowtie}$ logically imply every propagation rules implemented by the constraint in Y . Since $\bowtie$ is bijective, the theorem is valid when X and Y are reversed.

Theorem 3 Let f_Y be a propagator on Y , and c_X be a constraint on X . If $\models (D_{init} \wedge c_X \wedge \bowtie(C)) \rightarrow \bowtie(c)$ for all $(C \rightarrow c) \in prop(f_Y)$, then $\{dom(c_X)\} \cup F_{\bowtie} \gg \{f_Y\}$.

Example 4 Smith [2000] suggests two ways to model the Langford's problem as a permutation problem and how to combine them with the permutation channel. She points out that the so-called *minimal combined model*, which includes only X model and the permutation channel, gives as much pruning as the full combined model. This is proved in an ad hoc manner by Choi and Lee [2002]. We prove this formally using our generic approaches.²

The disequality constraints are propagation redundant by Lemma 2. Consider the separation constraints $c_Y \equiv y_j = 3i \Leftrightarrow y_{j+(i+2)} = 3i + 1$. The propagation rules for $dom(c_Y)$ are (r1) $y_j = 3i \rightarrow y_{j+(i+2)} = 3i + 1$, (r2) $y_{j+(i+2)} = 3i + 1 \rightarrow y_j = 3i$, (r3) $y_j \neq 3i \rightarrow y_{j+(i+2)} \neq 3i + 1$, and (r4) $y_{j+(i+2)} \neq 3i + 1 \rightarrow y_j \neq 3i$. We have that for $c_X \equiv x_{3i+1} = x_{3i} + (i+2)$, $\models (D_{init} \wedge c_X \wedge \bowtie(C)) \rightarrow \bowtie(c)$ for all propagation rules above. For example (r1) is mapped to $x_{3i} = j \rightarrow x_{3i+1} = j + i + 2$. Hence the conditions of Theorem 3 hold and $dom(c_Y)$ is propagation redundant.

²The complete description of the two permutation models for the Langford's Problem can be found in [Choi and Lee, 2002].

Note the importance of *equational* propagation rules such as $y_j = 3i \mapsto y_{j+(i+2)} = 3i + 1$. If we only allowed disequations on the right hand side, we would replace this with rules $y_j = 3i \mapsto y_{j+(i+2)} \neq k, 0 \leq k \neq 3i + 1 \leq n$. It is impossible to prove that the translated versions are implied by $D_{init} \wedge c_X$.

We can similarly show that each of the constraints $c'_Y \equiv y_j = 3i \Leftrightarrow y_{j+2(i+2)} = 3i + 2$ are propagation redundant using $c_X \wedge c'_X$, where $c'_X \equiv x_{3i+2} = x_{3i+1} + (i+2)$ by Theorem 3. Although model M_X does not include a domain propagator for $c_X \wedge c'_X$, we can still show propagation redundancy since $\{dom(c_X), dom(c'_X)\} \approx \{dom(c_X \wedge c'_X)\}$ by Lemma 1.

Similar reasoning applies to show that each constraint $y_j \neq 3i$ where $0 \leq i \leq 8$ and $27 - 2(i+2) \leq j \leq 26$ is made propagation redundant by $(x_{3i+1} = x_{3i} + (i+2)) \wedge (x_{3i+2} = x_{3i+1} + (i+2)) \wedge (0 \leq x_{3i+2} \leq 26)$. $\square$

4 Conclusion

We have extend our approach to other types of channelling constraints and lead to significantly faster models that do not increase the search space. Although we have illustrated the use of the theorems herein by hand, the approach can clearly be automated. We can constructs the propagation rules automatically using the approach of Abdennadher and Rigotti [2002]. We are interested in extending the work to reason bounds propagation. Another direction is to study a weaker notion of propagation redundancy which allows removal of constraints without affecting the search space given a specific search heuristic.

References

- [Abdennadher and Rigotti, 2002] S. Abdennadher and C. Rigotti. Automatic generation of rule-based solvers for intentionally defined constraints. *IJTAI*, 11(2):283–302, 2002.
- [Apt and Monfroy, 2001] K. Apt and E. Monfroy. Constraint programming viewed as rule-based programming. *TPLP*, 1(6):713–750, 2001.
- [Cheng et al., 1999] B.M.W. Cheng, K.M.F. Choi, J.H.M. Lee, and J.C.K. Wu. Increasing constraint propagation by redundant modelling: an experience report. *Constraints*, 4(2):167–192, 1999.
- [Choi and Lee, 2002] C.W. Choi and J.H.M. Lee. On the pruning behaviour of minimal combined models for permutation CSPs. In *International Workshop on Reformulating Constraint Satisfaction Problems*, 2002.
- [Schulte and Stuckey, 2001] C. Schulte and P. J. Stuckey. When do bounds and domain propagation lead to the same search space. In *PPDP 2001*, pages 115–126, 2001.
- [Smith, 2000] B. M. Smith. Modelling a permutation problem. Research Report 2000.18, School of Computer Studies, University of Leeds, 2000.
- [Walsh, 2001] T. Walsh. Permutation problems and channelling constraints. In *LPAR 2001*, pages 377–391, 2001.

Channeling Constraints and Value Ordering in the QuasiGroup Completion Problem

Iván Dotú and Alvaro del Val and Manuel Cebrián

Departamento de Ingeniería Informática
Universidad Autónoma de Madrid

1 Introduction

The Quasigroup Completion Problem (QCP) is a very challenging benchmark among combinatorial problems, and a focus of much recent interest in the area of constraint programming. It has numerous practical applications [Gomes *et al.*, 2002]; it has been put forward as a benchmark which can bridge the gap between purely random instances and highly structured problems [Gomes *et al.*, 2002]; and its structure as a multiple permutation problem [Walsh, 2001] is common to many other important problems in constraint satisfaction.

[Gomes *et al.*, 2002] 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 [Cheng *et al.*, 1996] with primal and dual models of the problem connected by channeling constraints [Walsh, 2001]– with some novel aspects, as well as a new and very effective value ordering heuristic.

2 Models for QCPs

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 *et al.*, 2002]. 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 defining a Latin Square, which must be filled with “colors” so that the colors of each row are all distinct, and similarly for columns. A QCP is the problem of coloring a partially filled Latin square, which is known to be NP-Complete [Colbourn, 1984].

Each row and column of a Latin Square defines a permutation problem, i.e. a constraint satisfaction problem with the same number of variables as values, where a solution is a permutation of the values [Walsh, 2001]. The QCP is a multiple permutation problem with $2n$ intersecting permutation constraints (n row permutation constraints and n column permutation constraints). In order to represent those permutation problems we have implemented three models:

- *Primal Model*: This model has a variable $x_{i,j}$ for each cell of the latin square, with i and j its coordinates. Their possible values are the colors of the cell.
- *Row Dual Model*: This is the first dual model, it represents the colors in each row; the dual variable $r_{i,k}$ is the k -th color in the i -th row, and the values it can take are the columns where it can be placed.
- *Column Dual Model*: This one models the colors in each column; the dual variable $c_{j,k}$ represents the k -th color in the j -th column, with rows as its possible values.

The primal and dual not-equal constraints specify that the colors in a row or column must all be different. While these are sufficient to capture the problem (and in fact redundantly so), it is known that these models can be improved by connecting them through *channeling constraints*:

- *Row Channeling Constraints*: Constraints for the n row permutation constraints, linking the primal model with the row dual model: $x_{i,j} = k \Leftrightarrow r_{i,k} = j$.
- *Column Channeling Constraints*: Corresponding to the n column permutation constraints, they link the primal and dual column models: $x_{i,j} = k \Leftrightarrow c_{j,k} = i$.
- *Triangular Channeling Constraints*: These (novel) constraints link both dual models, closing a “triangle” among the three models: $c_{j,k} = i \Leftrightarrow r_{i,k} = j$.

It’s easy to show that the first two kinds of channeling constraint fully capture a QCP (see also [Walsh, 2001]), making the primal and dual constraints redundant.

3 Value Ordering

Key to our results is a new *value ordering* heuristic, which we might call the *min-domain value selection heuristic*. 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 $x_{i,j}$. Then we choose a color k from its currently active domain for which the sum of the current domain sizes of $r_{i,k}$ and $c_{j,k}$ is minimal among the currently available colors for $x_{i,j}$.

4 Results

We focused our experiments on solvable instances. To make our results comparable with those reported in [Gomes *et al.*,

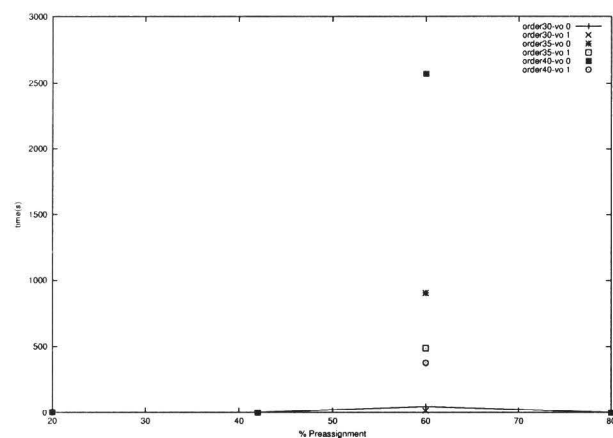


Figure 1: Mean solution time on QCPs of order 30, 35 and 40 with (vo1) and without value ordering (vo0).

2002], we used the same generator¹, and considered only “balanced” instances, which are known to be the hardest. We used an (AC3-based) Generalized Arc Consistency algorithm based on van Beek’s CPLAN implementation (<http://ai.uwaterloo.ca/~vanbeek/software/software.html>), with a few improvements. We considered various (complete) combinations of all the above mentioned constraints, and found that adding the primal and dual “not-equal” constraints never helped performance in the presence of row and column channeling constraints, and yielded even worse results in their absence. Most of the results discussed below therefore use a model with only row and column channeling constraints; they also use “MAC” (maintaining arc consistency [Bessiere, Regin, 1996]). Since our preliminary tests showed that CBJ and nogood learning seem to never help in these problems, and are often a source of significant overhead, we discarded them. We also considered various variable selection heuristics. We found the *min-domain* heuristic, which selects a variable with smallest domain to be uniformly the best among the alternatives we tried, which included more sophisticated heuristics such as *dom+degree*, *dom/degree*, and variants of the above which took into account the (primal or dual) model to which variables belong, e.g. selecting only among primal variables, etc.

With only row and column channeling constraints, our implementation outperforms classical approaches, in that we can solve a considerable amount of instances of order 40 in a reasonable time. As mentioned above, this was very recently considered as out of reach for pure constraint programming approaches [Gomes *et al.*, 2002], which ours is (in fact, we don’t even use global or non-binary constraints). The results when this model was used with the *min-domain* 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) it took 2905 secs without value ordering and only 0.40 secs with it. In Figure 1 we present a comparison of our results with and without value

¹Generator *lsencode*, kindly provided by Carla Gomes.

ordering for quasigroups of order 30,35 and 40, and 60% preassignment, close to the phase transition [Gomes *et al.*, 2002].

Encouraged by this performance, we generated some instances of order 45 and 60% preassignment. Again the value ordering heuristic turns out to be so efficient that we were able to solve a great number of instances in a reasonable amount of time. The next table shows median and mean time in seconds (the latter taken only over solved instances) and percent of solved instances, using value ordering for problems in the transition phase.

order	mean	median	% solved	timeout
30	148.84	174.11	68%	1000
35	533.43	163.48	84%	3600
40	690.85	842.11	68%	5000
45	1090.10	2971.40	56%	6000

Finally, we are currently exploring the effect of adding “triangular” channeling constraints; our preliminary results suggest that they can improve performance by an additional 30–35%, using in this case only forward checking (e.g. from 902 to 666 seconds in one order 45/60% preassigned instance). The idea is that they provably allow forward checking to perform as much pruning as arc-consistency, while saving many useless checks; in fact, as many as 60% fewer checks, even though the savings in time are not as great due to effects we are currently exploring.

5 Conclusions and Future Work

We have shown that a pure CSP approach can handle quasigroup completion problems significantly larger than was thought possible, using appropriate models and value ordering heuristics, and even in the absence of global alldiff constraints. Yet our results are preliminary, and no doubt it would be interesting to consider other value ordering heuristics and ways to make the triangular channeling constraints most effective.

References

- [Gomes *et al.*, 2002] C. Gomes and D. B. Shmoys. The promise of LP to Boost CSP Techniques for Combinatorial Problems. In *Proc. of CPAIOR’02*, Le Croisic, March 2002.
- [Gomes *et al.*, 2002] C. Gomes and D. B. Shmoys. Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem In *Proc. Computational Symposium on Graph Coloring and Generalizations*, 2002.
- [Walsh, 2001] T. Walsh. Permutation Problems and Channeling Constraints. TR 26, APES Group, 2001.
- [Colbourn, 1984] C. Colbourn. The complexity of completing partial latin squares. In *Discrete applied Mathematics*, (8):25–30, 1984.
- [Cheng *et al.*, 1996] 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.
- [Bessiere, Regin, 1996] C.Bessiere and J.C.Regina Mac and combined heuristics: two reason to forsake FC (and CBJ?) on hard problemss. In *CP’96*, pp. 61–75, Cambridge MA.