Automatic Generation of Redundant Models for Permutation Constraint Satisfaction Problems

Y.C. Law and J.H.M. Lee
Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{yclaw,jlee}@cse.cuhk.edu.hk

B.M. Smith
Cork Constraint Computation Centre,
University College Cork, Cork, Ireland
b.smith@4c.ucc.ie

Abstract

If we have two representations of a problem as constraint satisfaction problem (CSP) models, it has been shown that combining the models using channeling constraints can increase constraint propagation in tree search CSP solvers. Handcrafting two CSP models for a problem, however, is often time-consuming. In this paper, we propose *model induction*, a process which generates a second CSP model from an existing model using channeling constraints, and study its theoretical properties. The generated *induced model* is in a different viewpoint, i.e., set of variables. It is mutually redundant to and can be combined with the input model, so that the combined model contains more redundant information, which is useful to increase constraint propagation. We also propose two methods of combining CSP models, namely model intersection and model channeling. The two methods allow combining two mutually redundant models in the same and different viewpoints respectively. We exploit the applications of model induction, intersection, and channeling and identify three *new* classes of combined models, which contain different amounts of redundant information. We construct combined models of Permutation CSPs and show in extensive benchmark results that the combined models are more robust and efficient to solve than the single models.

1 Introduction

The task at hand is to tackle *constraint satisfaction problems* (CSPs), the definition of which, in the sense of Mackworth [29], can be stated as follows:

> *We are given a set of variables, a domain of possible values for each variable, and a conjunction of constraints. Each constraint is a rela-*

tion defined over a subset of the variables, limiting the combination of values that the variables in this subset can take. The goal is to find a consistent assignment of values to the variables so that all the constraints are satisfied simultaneously.

When tackling a real life problem, selecting the most appropriate CSP model for the problem is difficult in general. However, if we have two different formulations of the problem, we need not choose between them. Cheng *et al.* [6, 5] showed that two different models of a problem, based on different sets of variables, can be used at the same time; they introduced *channeling constraints* to link the variables of the two models. Each model, since it completely represents the problem being modeled, is (logically) redundant with respect to the other. However, using both models, linked by channeling constraints, can reduce the tree search efforts required to solve the problem, even though each provides essentially the same information about the problem.

Redundant modeling [6, 5] requires two CSP models of a problem. However, human modelers often find handcrafting CSP models time consuming. In this paper, we propose a process called *model induction* which automatically generates a CSP model from a given one using channeling constraints. The generated *induced model* has a different *viewpoint*, i.e., set of variables, from the original model. We give the syntactic construction rule for model induction and detailed examples, and prove its properties.

An induced model can be combined with other models of a problem to enjoy the benefits of reduced search effort in CSP solving. We show that two models of the same problem can share the same variables. We therefore propose *model intersection* and *model channeling* as two different methods to combine CSP models. Model intersection allows combining models with the same variables, while model channeling allows combining models with different variables, which is indeed a formalization of redundant modeling [5]. By exploiting different applications of model induction, intersection, and channeling, we identify three *new* classes of combined models, which contain different amount of redundant information. We construct different combined models of Permutation CSPs and show in our extensive benchmark results that the proposed classes of combined models are more robust and efficient to solve than the single models do.

This paper, a revised and extended version of the work by Law and Lee [25], is organized as follows. In Section 2, we present a brief review of related work about CSP models and reformulations, and combining models. Section 3 provides the background to the paper, where we formally introduce the *viewpoint* notion. Section 4 formally introduces model induction. We give its syntactic construction rule with detailed examples, examine its properties, and analyze its time and space complexities. In Section 5, we introduce model intersection and channeling. We exploit the applications of model induction and the two methods to give the three new classes of combined models. In Section 6, we present experimental results on combining mutually redundant models of Permutation CSPs using our proposed scheme applied to Langford's problem, random CSPs, and mutually redundant random CSP pairs. We conclude the paper in Section

7 by discussing our contributions and possible directions for future research.

2 Related Work

Rossi *et al.* [32] proposed a definition of equivalence of CSPs, based on the concept of mutual reducibility. Two CSPs are equivalent if it is possible to obtain the solutions of one CSP from those of another, and vice versa. Their definition makes a fundamental contribution in that it allows us to consider two different CSPs to be models of the same problem. Geelen [19] introduced two improved problem-independent value and variable ordering heuristics for solving CSPs and introduced a "dual-viewpoint" approach for a special but broad class of CSPs, namely Permutation CSPs. We can consider a Permutation CSP from two different perspectives. In one perspective, we solve the problem by finding a value for every variable, while in another perspective, we solve the problem by finding a variable for each value. Weigel and Bliek [39] introduced an algorithm to transform a CSP into its Boolean form which is then used to find reformulations. The resulting reformulations have different redundant constraints, and thus one may allow pruning in some situations where it is not possible in another.

Jourdan [23, 24] discussed multiple modeling, in which models representing different but redundant views of the same problem are synchronized using the communication mechanisms of constraint logic programming and concurrent constraint languages. Cheng *et al.* [6, 5] formally introduced redundant modeling. Two models of the same problem are combined together using *channeling constraints*. The combined model contains the two original models as sub-models. Channeling constraints allow the sub-models to cooperate during constraint solving by allowing constraint propagation to take place between them. This approach is shown to give increased constraint propagation and efficiency in a real life nurse rostering problem. Dotú *et al.* [13] used the redundant modeling technique to solve some hard Quasigroup Completion Problem instances in the phase transition region. Using an additional viewpoint and channeling constraints in CSP solving can also allow breaking value symmetries in matrix models efficiently [14, 26, 27].

Choi *et al.* [10, 11] introduced the notion of *propagation redundancy*. If a constraint is propagation redundant with respect to some constraints, that constraint cannot contribute extra pruning to reduce search effort. Hence, removing propagation redundant constraints will not sacrifice propagation but will save overhead for handling those constraints. Choi *et al.* gave general theorems about propagation redundancy of one constraint with respect to channeling constraints and constraints in the other model. Their work explains Smith's empirical results [34, 35] for Langford's problem that constraint propagation of the minimal combined model is the same as in redundant modeling. A minimal combined model is similar to redundant modeling but the constraints in the second model are dropped.

Hnich *et al.* [22] and Walsh [38] conducted extensive theoretical and empirical studies on using different models of Permutation CSPs and combined models using channeling constraints. They compared models by defining a measure of constraint tightness based on the level of consistency being enforced on the constraints in a model. Their results aid human CSP modelers to choose a viewpoint for Permutation CSPs. They also illustrated a general methodology for comparing different CSP models. Their works [22, 38] concentrate on the effect of different levels of constraint propagation on the constraints in a model to ensure a permutation for the variables in Permutation CSPs.

Another approach to automatic CSP model generation is to require human modelers to specify a problem in some formal language. Frisch *et al.* [17] proposed a modeling language ESSENCE, with some discrete mathematics notions, for specifying combinatorial problems in a manner similar to using natural human languages. A specification in ESSENCE can then be refined into a set of alternative CSP models using a system called CONJURE [18]. Flener *et al.* [15] designed the constraint language ESRA and showed how it is compiled into the programming language OPL. Hnich [21] introduced *function variables* which are variables whose domains are sets of functions. He showed how function variables and constraints can be mapped into different representations using integer and set variables, so that a model using function variables can automatically be transformed into alternate CSP models in terms of integer and set variables. Martínez-Hernández and Frisch [31] studied the automatic addition of redundant information and the corresponding channeling constraints to a CSP model.

CSP model transformation can be seen as a generalization of transformations from CSP to *satisfiability* (SAT). Walsh [37] comprehensively studied two CSP to SAT mappings, the *direct* and the *log* encodings, and compared the impact of maintaining arc consistency in a CSP with unit propagation on the SAT problem obtained from each of the mappings. Gent [20] carried out a similar comparison for the *support* encoding. Both Ansótegui and Manyà [3] and Frisch *et al.* [16] provided empirical evidence to assess the performance of the SAT problems obtained from CSP to SAT mappings. Frisch *et al.* [16] further proposed a new mapping to improve on the existing ones and identified sufficient conditions for omitting the at-least-one and at-most-one clauses to simplify the transformed problem.

3 Background

There is usually more than one way of formulating a problem P as a CSP. Central to the formulation process is to determine the variables and their domains (the sets of possible values the variables can take). Different choices of variables and domains result from viewing the problem P from different angles/perspectives. As in [25], we define a *viewpoint*[1] to be a pair (X, D), where $X = \{x_1, \ldots, x_n\}$ is a set of variables, and D is a set containing, for every $x \in X$, an associated domain $D(x)$ giving the set of possible values for x.

[1] Geelen [19] used the notion of viewpoint loosely but did not define it.

A viewpoint $V = (X, D)$ defines the possible assignments for variables in X. An *assignment* $x \mapsto a$ in a viewpoint V (or in a set of variables $U \subseteq X$) is a mapping from variable $x \in X$ (or U) to a value $a \in D(x)$. A *compound assignment* in V (or in $U \subseteq X$) is a set of assignments $\{x_{i_j} \mapsto a_j \mid 1 \leq j \leq k\}$, where $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ (or U) and $a_j \in D(x_{i_j})$ for $1 \leq j \leq k$. Note the requirement that no variable may be assigned more than one value in a compound assignment. A compound assignment θ in U can be projected onto $U' \subseteq U$ using $\pi_{U'}(\theta) = \{x \mapsto a \mid x \in U' \wedge (x \mapsto a) \in \theta\}$. Given a set of assignments θ, we use the predicate $cmpd(\theta, V)$ to ensure that θ is a compound assignment in V. A *complete assignment* in V is a compound assignment $\{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$ for all variables in X. Given a problem P and any viewpoint V, suppose $sol(P)$ is the set of all solutions of P (in whatever notation and formalism). We say that a V is a viewpoint *for problem* P if we can find a subset S of the set of all possible complete assignments in V so that there is a one-to-one mapping between S and $sol(P)$.

A *constraint* c places restrictions on a subset of variables limiting the combination of values that these variables can take. The subset of variables is called the *scope* of c, denoted by $var(c)$. We abuse terminology by saying that the compound assignment $\theta = \{x_{i_j} \mapsto a_j \mid 1 \leq j \leq k\}$ also has a scope $var(\theta) = \{x_{i_1}, \ldots, x_{i_k}\}$. Given a compound assignment θ such that $var(c) \subseteq var(\theta)$, we use $c\theta$ as a predicate to denote whether θ satisfies or violates c.

A *CSP* M is a pair (V, C), where V is a viewpoint and C is a set of constraints in V. A *solution* of M is a complete assignment θ in V so that $\bigwedge_{c \in C} c\theta$ is true. A CSP M is *satisfiable* if $sol(M) \neq \emptyset$, and *unsatisfiable* otherwise. We say that a CSP $M = (V, C)$ is a *model* for a problem P if V is a viewpoint for P and the constraints C in M are defined in such a way that the one-to-one correspondence is maintained (i.e., $S = sol(M)$).

Suppose M_1 and M_2 are two different models for the same problem P. By definition, there exists a one-to-one mapping between $sol(M_1)$ and $sol(M_2)$. We say that M_1 and M_2 are *mutually redundant*. As we shall see, it is possible for mutually redundant models M_1 and M_2 to have the same viewpoint. In that special case, it is easy to verify that $sol(M_1) = sol(M_2)$.

Given two models $M_1 = ((X, D_X), C_X)$ and $M_2 = ((Y, D_Y), C_Y)$ for the same problem, Cheng *et al.* [5] defined a *channeling constraint* c to be a constraint where $var(c) \not\subseteq X$, $var(c) \not\subseteq Y$, and $var(c) \subseteq X \cup Y$, and c relates M_1 and M_2 by limiting the combination of values that their variables can take. Note that in the definition of channeling constraints, the constraints in the two models are immaterial. Channeling constraints relate only the viewpoints of the models; they set forth a relationship between the possible assignments of the two viewpoints.

We illustrate the ideas of channeling constraints and mutual redundancy of models using the 4-queens problem, which is to place four queens on a 4×4 chessboard such that no two queens can attack each other. A textbook model $M_1 = ((X, D_X), C_X)$ of the problem has four variables $X = \{x_1, x_2, x_3, x_4\}$. Each x_i denotes the column position of queen i in row i of the chessboard. The domain of the variables is thus $D_X(x_i) = \{1, 2, 3, 4\}$ for $1 \leq i \leq 4$. The choice of variables ensures no two queens can be on the same row by default. The constraints C_X enforce that no two queens can be on the same column or diagonal:

- $x_i \neq x_j$ for $1 \leq i < j \leq 4$; and
- $|x_i - x_j| \neq j - i$ for $1 \leq i < j \leq 4$.

The two solutions of this model are $\theta_1 = \{x_1 \mapsto 2, x_2 \mapsto 4, x_3 \mapsto 1, x_4 \mapsto 3\}$ and $\theta_2 = \{x_1 \mapsto 3, x_2 \mapsto 1, x_3 \mapsto 4, x_4 \mapsto 2\}$.

The 4-queens problem can be modeled in another viewpoint (Y, D_Y). which also has four variables $Y = \{y_1, y_2, y_3, y_4\}$. Each y_j denotes the row position of queen j in column j of the chessboard. The domain of the variables is thus $D_Y(y_j) = \{1, 2, 3, 4\}$ for $1 \leq j \leq 4$. The constraints C_Y can be stated in this viewpoint accordingly, giving the model $M_2 = ((Y, D_Y), C_Y)$. The two solutions of this model are $\phi_1 = \{y_1 \mapsto 3, y_2 \mapsto 1, y_3 \mapsto 4, y_4 \mapsto 2\}$ and $\phi_2 = \{y_1 \mapsto 2, y_2 \mapsto 4, y_3 \mapsto 1, y_4 \mapsto 3\}$.

To connect the models M_1 and M_2, we can state the channeling constraints as $x_i = j \Leftrightarrow y_j = i$ for $1 \leq i, j \leq 4$. They collectively define a one-to-one mapping between $sol(M_1)$ and $sol(M_2)$ such that θ_1 corresponds to ϕ_1 and θ_2 corresponds to ϕ_2. The two models M_1 and M_2 are mutually redundant to each other.

4 Model Induction

In this section, we introduce *model induction*: a method for systematically generating a new *induced model* from an existing model, using another viewpoint and channeling constraints as input. Model induction is a transformation of the constraints of one model, both implicit and explicit, to constraints in another viewpoint. In the following, we describe the choice of channeling constraints, the construction of induced models with detailed examples, and the properties and complexities of model induction.

4.1 Channeling Constraints

In order for model induction to generate a mutually redundant model, the set of channeling constraints cannot be chosen arbitrarily. Given viewpoints $V_X = (X, D_X)$ and $V_Y = (Y, D_Y)$, suppose we want to construct an induced model in V_Y using a model in V_X. A necessary condition is that the set of channeling constraints between V_X and V_Y must collectively define a *total* function f from the possible assignments in V_X to those in V_Y:

$$f : \{x \mapsto a \,|\, x \in X \wedge a \in D_X(x)\} \rightarrow \{y \mapsto b \,|\, y \in Y \wedge b \in D_Y(y)\}.$$

In other words, f maps *every* assignment in V_X to an assignment in V_Y. We call the function f a *channeling function*. We overload a channeling function f to act also on a set of assignments θ such that $f(\theta) = \{f(x \mapsto a) \,|\, (x \mapsto a) \in \theta\}$.

Note that the channeling *constraints* are used to connect two models and propagate pruning information between the two models. The concept of a channeling *function*, however, is introduced solely for the purpose of generating a model from one viewpoint to another, which is a one way process. Therefore, the channeling function does not have to be bijective. We shall see later examples of model induction using both injective and bijective channeling functions.

4.2 Induced Models

The description of model induction assumes that constraints are represented extensionally. We define $ng(c)$ of a constraint c as the set of no-goods in $var(c)$. Suppose $var(c) = \{x_{i_1}, \ldots, x_{i_k}\}$, a *no-good* $\{x_{i_1} \mapsto a_1, \ldots, x_{i_k} \mapsto a_k\}$ of c is a compound assignment in $var(c)$ that violates c. It has the logical meaning $\neg((x_{i_1} = a_1) \wedge \ldots \wedge (x_{i_k} = a_k))$.

Given a CSP $M = (V_X, C_X)$ where $V_X = (X, D_X)$, a viewpoint $V_Y = (Y, D_Y)$, and a channeling function f from the possible assignments in V_X to those in V_Y, we note that M contains two types of constraints: the *explicit* constraints stated in C_X and the implicit constraints on variable assignments. The implicit constraints are similar to the uniqueness constraints by Weigel and Bliek [39] and can be further broken down into the *No-Double-Assignment* (NDA) and *At-Least-One-Assignment* (ALOA) constraints. They restrict each variable to be assigned no more than one value and at least one value from its domain respectively. The idea of model induction is to use f to transform both the explicit and the implicit constraints in M to constraints C_Y in viewpoint V_Y, yielding the induced model $i(f, M) = (V_Y, C_Y)$.

We first transform the explicit constraints stated in C_X. Since a constraint in C_X is represented by a set of no-goods, we can apply f to every assignment in each no-good of all constraints in C_X to collect the transformed no-goods in a set S_Y:

$$S_Y = \{f(\theta) \mid c \in C_X \wedge \theta \in ng(c) \wedge cmpd(f(\theta), V_Y)\}.$$

It is indeed possible for $f(\theta)$ not to be a valid compound assignment. For instance, suppose $\theta = \{x_1 \mapsto 1, x_2 \mapsto 2\}$, $f(x_1 \mapsto 1) = y \mapsto 1$, and $f(x_2 \mapsto 2) = y \mapsto 2$. The set $f(\theta) = \{y \mapsto 1, y \mapsto 2\}$ is *not* a compound assignment. The viewpoint V_Y already restricts implicitly that the variable y cannot be assigned values 1 and 2 simultaneously. Therefore, we can discard $f(\theta)$.

Example 1 *Suppose $c \in C_X$ and $\{x_1 \mapsto 1, x_2 \mapsto 2\} \in ng(c)$. If $f(x_1 \mapsto 1) = y_2 \mapsto 1$ and $f(x_2 \mapsto 2) = y_1 \mapsto 1$, then $f(\{x_1 \mapsto 1, x_2 \mapsto 2\}) = \{y_1 \mapsto 1, y_2 \mapsto 1\}$ is in S_Y. Further suppose $\{x_1 \mapsto 1, x_2 \mapsto 1\} \in ng(c)$ and $f(x_2 \mapsto 1) = y_2 \mapsto 2$. Then $f(\{x_1 \mapsto 1, x_2 \mapsto 1\}) = \{y_2 \mapsto 1, y_2 \mapsto 2\}$ is not a compound assignment and is not in S_Y.* □

Implicit in a CSP formulation is that each variable should be assigned exactly one value. Part of this restriction can be translated to the NDA constraints which require that no variable can be assigned two values from its domain at the same time. This corresponds to a set of (invalid) no-goods of the form

$\{x \mapsto a, x \mapsto b\}$ for all $x \in X$ and $a, b \in D_X(x)$ (where $a \neq b$), which is satisfied implicitly and not represented in M. Their transformed counterparts, however, are needed in the induced model. We apply f to every assignment in the invalid no-goods, and collect the valid transformations in a set N_Y:

$$N_Y = \bigcup_{x \in X} \{\theta \mid a, b \in D_X(x) \wedge a \neq b \wedge \theta = f(\{x \mapsto a, x \mapsto b\}) \wedge \mathit{cmpd}(\theta, V_Y)\}.$$

Example 2 *Suppose* $D_X(x_1) = \{2, 3\}$*,* $f(x_1 \mapsto 2) = y_1 \mapsto 2$*, and* $f(x_1 \mapsto 3) = y_2 \mapsto 1$*. Then,* $f(\{x_1 \mapsto 2, x_1 \mapsto 3\}) = \{y_1 \mapsto 2, y_2 \mapsto 1\}$ *is in* N_Y*.* $\Box$

The other type of implicit constraint in M can be translated to the ALOA constraints which require that each variable must be assigned at least one value from its domain. This corresponds to the constraints $\bigvee_{a \in D_X(x)} x = a$ for each $x \in X$, which are satisfied implicitly and not stated in M. Unlike the explicit and NDA constraints, these unary constraints cannot be expressed in terms of no-goods. For each $x \in X$, we first apply f to all the assignments of x. Let $D_X(x) = \{a_1, \ldots, a_d\}$. We obtain the assignments $f(x \mapsto a_1) = y_{j_1} \mapsto b_1, \ldots, f(x \mapsto a_d) = y_{j_d} \mapsto b_d$. The unary constraint $\bigvee_{1 \leq i \leq d} x = a_i$ is then logically equivalent to $\bigvee_{1 \leq i \leq d} y_{j_i} = b_i$ in V_Y, whose scope is $U = \{y_{j_1}, \ldots, y_{j_d}\}$. A compound assignment with scope U that contains any of $y_{j_i} \mapsto b_i$ for $1 \leq i \leq d$ will satisfy the constraint $\bigvee_{1 \leq i \leq d} y_{j_i} = b_i$. Therefore, the no-goods of the constraint are those compound assignments θ with $\mathit{var}(\theta) = U$ such that $\bigwedge_{1 \leq i \leq d} (y_{j_i} \mapsto b_i) \notin \theta$.

$$\begin{aligned} A_Y \quad = \quad & \bigcup_{x \in X} \{\theta \mid D_X(x) = \{a_1, \ldots, a_d\} \wedge (\bigwedge_{1 \leq i \leq d} f(x \mapsto a_i) = y_{j_i} \mapsto b_i) \wedge \\ & \qquad \mathit{var}(\theta) = \{y_{j_1}, \ldots, y_{j_d}\} \wedge \mathit{cmpd}(\theta, V_Y) \wedge \bigwedge_{1 \leq i \leq d} (y_{j_i} \mapsto b_i) \notin \theta\}. \end{aligned}$$

Example 3 *Suppose* $D_X(x_1) = D_Y(y_1) = D_Y(y_2) = \{1, 2, 3\}$*,* $f(x_1 \mapsto 1) = y_1 \mapsto 1$*,* $f(x_1 \mapsto 2) = y_1 \mapsto 2$*, and* $f(x_1 \mapsto 3) = y_2 \mapsto 1$*. Then, for variable* x_1*, applying* f *to the assignments* $x_1 \mapsto 1$*,* $x_1 \mapsto 2$*, and* $x_1 \mapsto 3$ *tells us that the no-goods from the ALOA constraint for* x_1 *have the scope* $\{y_1, y_2\}$*. All compound assignments* $\{y_1 \mapsto a, y_2 \mapsto b\}$ *that do not contain* any *of* $y_1 \mapsto 1$*,* $y_1 \mapsto 2$*, and* $y_2 \mapsto 1$ *are no-goods in* A_Y*. Thus,* $\{y_1 \mapsto a, y_2 \mapsto b\}$ *where* $a = 3$ *and* $2 \leq b \leq 3$*, i.e.,* $\{y_1 \mapsto 3, y_2 \mapsto 2\}$ *and* $\{y_1 \mapsto 3, y_2 \mapsto 3\}$*, are included in* A_Y*.* $\Box$

We note that the no-goods in a constraint $c \in C_X$ can be transformed to contribute to the no-goods of more than one constraint in the induced model. The set $S_Y \cup N_Y \cup A_Y$ consists of all the induced no-goods, which have different scopes in V_Y. The next step is to collect no-goods with the same scope from $S_Y \cup N_Y \cup A_Y$ to form a constraint in V_Y. Thus,

$$C_Y = \{c \mid \mathit{var}(c) \subseteq Y \wedge ng(c) = \sigma_{\mathit{var}(c)}(S_Y \cup N_Y \cup A_Y) \neq \emptyset\},$$

where $\sigma_U(\Theta) = \{\theta \mid \theta \in \Theta \wedge \mathit{var}(\theta) = U\}$.

4.3 Examples

We illustrate the construction of two induced models using the model M_1 of the 4-queens problem introduced in Section 3. The two examples show two common usages of model induction: the first transforms a model to the Boolean viewpoint (a viewpoint whose variables have Boolean domains), while the second demonstrates a transformation that is generally applicable to Permutation CSPs.

4.3.1 Integer-to-Boolean Induction

We first give an example which transforms M_1 to a model with Boolean variables (variables with $\{0,1\}$ domain). For the 4-queens problem, consider the Boolean viewpoint (Z, D_Z) with 16 variables $Z = \{z_{ij} \mid 1 \leq i, j \leq 4\}$ and $D_Z(z_{ij}) = \{0,1\}$ for $1 \leq i, j \leq 4$. The assignment $z_{ij} \mapsto 1$ denotes that chessboard position (i,j) (row i and column j) contains a queen; and $z_{ij} \mapsto 0$ denotes the contrary. The channeling constraints $x_i = j \Leftrightarrow z_{ij} = 1$ for $1 \leq i, j \leq 4$ define the channeling function:

$$g(x_i \mapsto j) = z_{ij} \mapsto 1 \text{ for } 1 \leq i, j \leq 4.$$

This channeling function is injective but not bijective, since it never generates assignments of the form "$z_{ij} \mapsto 0$."

We first transform the explicit constraints in C_X. The no-goods for the diagonal constraints in M_1 have the form $\{x_i \mapsto k, x_j \mapsto k \pm (i-j)\}$ for $1 \leq i < j \leq 4$, $1 \leq k \leq 4$, and $1 \leq k \pm (i-j) \leq 4$. Hence, the induced no-goods are $\{z_{ik} \mapsto 1, z_{j,k\pm(i-j)} \mapsto 1\}$. For example, the diagonal constraint $|x_1 - x_2| \neq 2-1$ generates the no-goods

$$\begin{array}{c}\{z_{11} \mapsto 1, z_{22} \mapsto 1\}, \{z_{12} \mapsto 1, z_{23} \mapsto 1\}, \{z_{13} \mapsto 1, z_{24} \mapsto 1\}, \\ \{z_{12} \mapsto 1, z_{21} \mapsto 1\}, \{z_{13} \mapsto 1, z_{22} \mapsto 1\}, \{z_{14} \mapsto 1, z_{23} \mapsto 1\}\end{array}$$

for inclusion in S_Z. The no-goods for the column constraints in M_1 have the form $\{x_i \mapsto k, x_j \mapsto k\}$ for $1 \leq i < j \leq 4$ and $1 \leq k \leq 4$. Hence, the induced no-goods are $\{z_{ik} \mapsto 1, z_{jk} \mapsto 1\}$. For example, the constraint $x_1 \neq x_2$ generates the no-goods

$$\{z_{11} \mapsto 1, z_{21} \mapsto 1\}, \{z_{12} \mapsto 1, z_{22} \mapsto 1\}, \{z_{13} \mapsto 1, z_{23} \mapsto 1\}, \{z_{14} \mapsto 1, z_{24} \mapsto 1\}$$

for inclusion in S_Z.

We then transform to (Z, D_Z) the NDA constraints which restrict each $x_i \in X$ not to be assigned two different values. Thus,

$$\begin{aligned} N_Z &= \bigcup_{x_i \in X} \{\{g(x_i \mapsto j_1), g(x_i \mapsto j_2)\} \mid 1 \leq j_1 < j_2 \leq 4\} \\ &= \{\{z_{ij_1} \mapsto 1, z_{ij_2} \mapsto 1\} \mid 1 \leq i \leq 4 \wedge 1 \leq j_1 < j_2 \leq 4\}. \end{aligned}$$

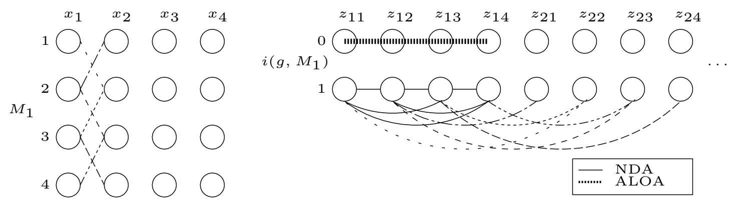


Figure 1: The Constraint $|x_1 - x_2| \neq 1$ in M_1 and its Induced Counterpart in $i(g, M_1)$ with the NDA and ALOA Constraints from x_1

For example, the NDA constraint for $x_1 \in X$ generates the no-goods

$$\{z_{11} \mapsto 1, z_{12} \mapsto 1\}, \{z_{11} \mapsto 1, z_{13} \mapsto 1\}, \{z_{11} \mapsto 1, z_{14} \mapsto 1\},$$
$$\{z_{12} \mapsto 1, z_{13} \mapsto 1\}, \{z_{12} \mapsto 1, z_{14} \mapsto 1\}, \{z_{13} \mapsto 1, z_{14} \mapsto 1\}$$

to ensure that no two queens will be placed on row 1 of the chessboard.

Last but not least, we need to transform the ALOA constraints. For each $x_i \in X$, applying g to the assignments $x_i \mapsto j$ for $1 \leq j \leq 4$ tells us that the no-goods in (Z, D_Z) have scope $\{z_{i1}, \ldots, z_{i4}\}$. The no-goods are those compound assignments $\{z_{i1} \mapsto q_1, \ldots, z_{i4} \mapsto q_4\}$ that do not contain *any* of $z_{i1} \mapsto 1$, $z_{i2} \mapsto 1$, $z_{i3} \mapsto 1$, and $z_{i4} \mapsto 1$. Since the domain of all variables z_{ij} is only $\{0, 1\}$, $\{z_{i1} \mapsto 0, \ldots, z_{i4} \mapsto 0\}$ is the only no-good needed. Thus:

$$\begin{aligned} A_Z \quad = \quad & \{\{z_{11} \mapsto 0, z_{12} \mapsto 0, z_{13} \mapsto 0, z_{14} \mapsto 0\}, \\ & \{z_{21} \mapsto 0, z_{22} \mapsto 0, z_{23} \mapsto 0, z_{24} \mapsto 0\}, \\ & \{z_{31} \mapsto 0, z_{32} \mapsto 0, z_{33} \mapsto 0, z_{34} \mapsto 0\}, \\ & \{z_{41} \mapsto 0, z_{42} \mapsto 0, z_{43} \mapsto 0, z_{44} \mapsto 0\}\}, \end{aligned}$$

meaning that a row of the chessboard cannot be empty.

The induced model $i(g, M_1) = ((Z, D_Z), C_Z)$ can be formed by extracting and grouping no-goods of the same scope to form a constraint in C_Z. As we shall see, by Theorem 1, $i(g, M_1)$ and M_1 are mutually redundant, and are both models of the 4-queens problem. Figure 1 shows the constraint $|x_1 - x_2| \neq 1$ in M_1 and its induced counterpart in $i(g, M_1)$ with the transformed NDA and ALOA constraints from variable x_1. In the figure, the nodes are the assignments and the edges are the no-goods between two nodes. In particular, the edge for the transformed ALOA constraint is a hyper-edge among the variables z_{11}, z_{12}, z_{13}, and z_{14}, denoting the no-good $\{z_{11} \mapsto 0, z_{12} \mapsto 0, z_{13} \mapsto 0, z_{14} \mapsto 0\}$.

This example shows that model induction can be used as an encoding of a CSP to a satisfiability (SAT) problem. In particular, it corresponds to the direct encoding [37] (also known as the standard mapping [3] and the unary/unary transform [16]). The NDA constraints are transformed to the at-most-one clauses (AMO) [3, 33, 16], while the ALOA constraints are transformed to the at-least-one clauses (ALO) [3, 33, 16].

Note that in our transformation of the NDA constraints, the number of no-goods generated per variable is quadratic in the domain size, and the arity of the no-goods is always two. In fact, it is possible to transform the NDA constraints in other ways. Roussel [33] proposed a different encoding of the NDA constraints, interesting for large domains. The encoding uses only a linear number of no-goods, but of a higher arity (which depends on the domain size). We choose instead to generate a larger number of low-arity constraints, since propagating high-arity constraints is less efficient in CSP solvers.

4.3.2 Permutation Induction

The model M_1 of the 4-queens problem is an example of a Permutation CSP. In a *Permutation CSP* [19, 34, 35, 22] $((X, D_X), C)$, we always have $D_X(x_i) = D_X(x_j)$ for $x_i, x_j \in X$, and $|D_X(x_i)| = |X|$. In addition, any solution $\{x_1 \mapsto k_1, \ldots, x_n \mapsto k_n\}$ of a Permutation CSP must have the property that $k_i \neq k_j$ for $1 \leq i < j \leq n$. For a Permutation CSP, a second viewpoint (Y, D_Y) always exists by interchanging the roles of variables and values. For example, in the 4-queens problem, we can have another viewpoint (Y, D_Y) for the 4-queens problem for model induction with $Y = \{y_1, y_2, y_3, y_4\}$ and $D_Y(y_j) = \{1, 2, 3, 4\}$ for $1 \leq j \leq 4$. Each y_j denotes the row position of the queen on column j. The channeling constraints $x_i = j \Leftrightarrow y_j = i$ for $1 \leq i, j \leq 4$ define the channeling function

$$f(x_i \mapsto j) = y_j \mapsto i \text{ for } 1 \leq i, j \leq 4.$$

Note that this channeling function is bijective, as we have $f^{-1}(y_j \mapsto i) = x_i \mapsto j$ for $1 \leq i, j \leq 4$. This is generally true for channeling functions for Permutation CSPs.

To obtain an induced model in the viewpoint (Y, D_Y), we first transform the explicit constraints in C_X. Recall the no-goods $\{x_i \mapsto k, x_j \mapsto k \pm (i-j)\}$ for the diagonal constraints in M_1. The induced no-goods are thus $\{y_k \mapsto i, y_{k \pm (i-j)} \mapsto j\}$. For example, the constraint $|x_1 - x_2| \neq 2 - 1$ generates the no-goods

$$\begin{array}{l}\{y_1 \mapsto 1, y_2 \mapsto 2\}, \{y_2 \mapsto 1, y_3 \mapsto 2\}, \{y_3 \mapsto 1, y_4 \mapsto 2\}, \\ \{y_2 \mapsto 1, y_1 \mapsto 2\}, \{y_3 \mapsto 1, y_2 \mapsto 2\}, \{y_4 \mapsto 1, y_3 \mapsto 2\}\end{array}$$

for inclusion in S_Y. For the column constraints, the no-goods in M_1 are $\{x_i \mapsto k, x_j \mapsto k\}$ for $1 \leq i < j \leq 4$ and $1 \leq k \leq 4$. Hence, the induced counterparts are $\{y_k \mapsto i, y_k \mapsto j\}$, which are *not* valid no-goods because they contain two different assignments for the same variable y_k. Thus, S_Y does not contain any no-goods corresponding to the column constraints.

The NDA constraints ensure that each $x_i \in X$ cannot be assigned two different values. They are transformed to the no-goods:

$$\begin{aligned} N_Y &= \bigcup_{x_i \in X} \{\{f(x_i \mapsto j_1), f(x_i \mapsto j_2)\} \mid 1 \leq j_1 < j_2 \leq 4\} \\ &= \{\{y_{j_1} \mapsto i, y_{j_2} \mapsto i\} \mid 1 \leq i \leq 4 \land 1 \leq j_1 < j_2 \leq 4\} \end{aligned}$$

in (Y, D_Y). For example, the NDA constraint for $x_1 \in X$ generates the no-goods

$$\begin{array}{l}\{y_1 \mapsto 1, y_2 \mapsto 1\}, \{y_1 \mapsto 1, y_3 \mapsto 1\}, \{y_1 \mapsto 1, y_4 \mapsto 1\}, \\ \{y_2 \mapsto 1, y_3 \mapsto 1\}, \{y_2 \mapsto 1, y_4 \mapsto 1\}, \{y_3 \mapsto 1, y_4 \mapsto 1\}\end{array}$$

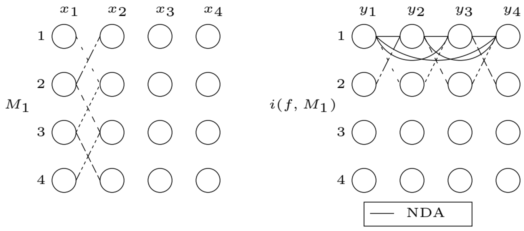


Figure 2: The constraint $|x_1 - x_2| \neq 1$ in M_1 and its induced counterpart in $i(f, M_1)$ with the transformed NDA constraints from x_1

to ensure that row 1 of the chessboard cannot contain two queens. In fact, the no-goods generated from the NDA constraints ensures that the induced model is also a Permutation CSP, since they guarantee that the values taken by the variables in Y must be all different in a solution.

The ALOA constraints are obtained from the implicit constraints "each $x_i \in X$ must be assigned at least one value" in M_1. For each $x_i \in X$, applying f to the assignments $x_i \mapsto j$ for $1 \leq j \leq 4$ tells us that the no-goods in (Y, D_Y) have the scope $\{y_1, \ldots, y_4\}$. The no-goods are those $\{y_1 \mapsto q_1, \ldots, y_4 \mapsto q_4\}$ that do not contain *any* of $y_1 \mapsto i$, $y_2 \mapsto i$, $y_3 \mapsto i$, and $y_4 \mapsto i$. Since the domain of all variables in Y is $\{1, 2, 3, 4\}$, we obtain the no-goods $\{y_1 \mapsto q_1, \ldots, y_4 \mapsto q_4\}$ for $1 \leq q_j \leq 4$, $1 \leq j \leq 4$, and $q_j \neq i$. For example, the no-goods for x_1 are $\{y_1 \mapsto q_1, \ldots, y_4 \mapsto q_4\}$ for $2 \leq q_1, q_2, q_3, q_4 \leq 4$. The intuitive meaning of these no-goods is that a row of the chessboard cannot be empty. However, the NDA constraints ensure that at most one queen can be placed in each of the four rows. Furthermore, we have to place four queens in the chessboard. Therefore, there can never be no queens in a row of the chessboard, and the information from these no-goods is implied already. Hence, we can ignore the ALOA constraints in this special case and allow $A_Y = \emptyset$. In the rest of the paper, we assume that whenever we perform model induction on one Permutation CSP to obtain another Permutation CSP using the channeling function $f(x_i \mapsto j) = y_j \mapsto i$, we set $A_Y = \emptyset$.

The induced model $i(f, M_1) = ((Y, D_Y), C_Y)$ can be formed by extracting and grouping no-goods of the same scopes to form constraints in C_Y. In this special example, the induced model $i(f, M_1)$ is equal to the model M_2 given in Section 3. The models $i(g, M_1)$, $i(f, M_1)$, and M_1 are mutually redundant to each other, and all of them are models of the 4-queens problem. Figure 2 shows the constraint $|x_1 - x_2| \neq 1$ in M_1 and its induced counterpart in $i(f, M_1)$ with the transformed NDA constraint from variable x_1.

This example shows that given a Permutation CSP and the channeling function $f(x_i \mapsto j) = y_j \mapsto i$, model induction can always be used to generate a redundant model which is also a Permutation CSP.

4.4 Properties

The following theorem and corollaries give an important consequence of model induction: the transformation of no-goods is meaning-preserving. Theorem 1 states that applying model induction on M generates a mutually redundant model $i(f, M)$ of M.

Theorem 1 *If $M = (V_1, C_1)$ is a model for problem P and V_2 is a viewpoint for P, then M and $i(f, M)$ are mutually redundant models for all total functions f (defined by channeling constraints connecting V_1 and V_2) mapping from possible assignments in V_1 to those in V_2.*

Proof 1 *Suppose there is a set of channeling constraints defining a total function f such that M and $i(f, M)$ are not mutually redundant. By definition, there is no one-to-one mapping between $sol(M)$ and $sol(i(f, M))$, nor between $sol(P)$ and $sol(i(f, M))$ (since M is a model for P). Thus no subset S of the set of all possible complete assignments in V_2 can have a one-to-one mapping with $sol(i(f, M))$. Consequently, V_2 cannot be a viewpoint for P; hence a contradiction.* $\Box$

Given M_1 and M_2 in viewpoints V_1 and V_2 respectively, we can generate an induced model using M_1 that has the same solution set as M_2.

Corollary 2 *If $M_1 = (V_1, C_1)$ and $M_2 = (V_2, C_2)$ are mutually redundant models for P, and f is a total function mapping from possible assignments in V_1 to those in V_2, then $sol(M_2) = sol(i(f, M_1))$.*

Proof 2 *By Theorem 1, M_1 and $i(f, M_1)$ are mutually redundant. So are M_1, M_2, and $i(f, M_1)$. By definition, there must be a one-to-one mapping between $sol(M_2)$ and $sol(i(f, M_1))$. Since M_2 and $i(f, M_1)$ have the same viewpoint V_2, the only possible one-to-one mapping is the identity mapping. Thus, $sol(M_2) = sol(i(f, M_1))$.* $\Box$

Inducing a model twice using f and f^{-1} generates a model that has the same solution set as the original model.

Corollary 3 *If $M = (V_1, C_1)$ is a model for problem P, V_2 is a viewpoint for P, and f is a total and bijective function (i.e., f^{-1} exists) mapping from possible assignments in V_1 to those in V_2, then $sol(i(f^{-1}, i(f, M))) = sol(M)$.*

Proof 3 *By Theorem 1, M and $i(f, M)$ are mutually redundant. Similarly, $i(f, M)$ and $i(f^{-1}, i(f, M))$ are mutually redundant. M and $i(f^{-1}, i(f, M))$ have the same viewpoint. By Corollary 2, $sol(i(f^{-1}, i(f, M))) = sol(M)$.* $\Box$

In general, M and $i(f^{-1}, i(f, M))$ are not identical, although they share the same viewpoint and the same solution set; applying model induction twice can result in adding (logically) redundant information to M. However, applying model induction twice on a Permutation CSP using channeling function f and its inverse f^{-1}, where $f(x_i \mapsto j) = y_j \mapsto i$ for all i, j, can result in the same CSP as the original. The following theorems and corollaries state when model induction cannot obtain new models for Permutation CSPs.

Theorem 4 *If the following conditions hold:*

1. $M = ((X, D), C)$ *is a Permutation CSP;*
2. *No two constraints in* M *have the same scope;*
3. *The all-different constraints in* M *are ensured by the no-goods* $\{x_i \mapsto k, x_j \mapsto k\}$ *for* $x_i, x_j \in X$ *and* $k \in D(x_i)$ *with* $i \neq j$*;*
4. *There are not any no-goods* θ *in the constraints of* M *such that for* $x_i, x_j \in X$ $(i \neq j)$, $\pi_{\{x_i,x_j\}}(\theta) = \{x_i \mapsto k, x_j \mapsto k\}$ *except those in (3); and*
5. *We have another viewpoint connected with channeling constraints defining* $f(x_i \mapsto j) = y_j \mapsto i$ *for all* i, j*,*

then

$$M = i(f^{-1}, i(f, M)).$$

Proof 4 *Since no two constraints in* M *have the same scope, we can consider the set of no-goods of all constraints in* M*. Let* $\Theta = \{\theta \mid c \in C \wedge \theta \in ng(c)\}$ *be such set of no-goods. Alternatively, we can write* $\Theta = \Theta_{ad} \cup \Theta_s$*, where*

$$\begin{aligned} \Theta_{ad} &= \{\{x_i \mapsto k, x_j \mapsto k\} \mid x_i, x_j \in X \wedge k \in D(x_i) \wedge i \neq j\} \text{ and} \\ \Theta_s &= \{\theta \mid c \in C \wedge \theta \in ng(c) \wedge \theta \notin \Theta_{ad}\}. \end{aligned}$$

Θ_{ad} *is the set of the no-goods of the all-different constraints, while* Θ_s *is the set of other no-goods of the model. By condition (4), for all* $\theta \in \Theta_s$*,* $x_i, x_j \in X$*, and* $i \neq j$*, we do not have* $\pi_{\{x_i,x_j\}}(\theta) = \{x_i \mapsto k, x_j \mapsto k\}$*. Let* V *and* V' *be the viewpoints of* M *and* $i(f, M)$ *respectively. We first construct the set* Θ' *of all no-goods of* $i(f, M)$*.*

$$\begin{aligned} \Theta' = S_Y \cup N_Y &= \{f(\theta) \mid \theta \in \Theta \wedge cmpd(f(\theta), V')\} \cup N_Y \\ &= \{f(\theta) \mid \theta \in \Theta_s \cup \Theta_{ad} \wedge cmpd(f(\theta), V')\} \cup N_Y \\ &= \{f(\theta) \mid \theta \in \Theta_s\} \cup N_Y \end{aligned}$$

S_Y *and* N_Y *are the sets of no-goods in* $i(f, M)$ *transformed from the explicit constraints in* M *and the NDA constraints respectively. The no-goods transformed from the ALOA constraints are discarded since* $i(f, M)$ *involves model induction from a Permutation CSP to another. On the one hand, all no-goods in* Θ_s *can be transformed and included in* Θ' *since the individual assignments in a no-good in* Θ_s *must have different values in them. On the other hand, all no-goods in* Θ_{ad} *have the form* $\{x_i \mapsto k, x_j \mapsto k\}$*. Applying* f *to these no-goods yields* $\{y_k \mapsto i, y_k \mapsto j\}$ *which are invalid no-goods in* V' *and thus cannot be included in* Θ'*.*

With Θ', we can construct the set Θ'' of all no-goods in $i(f^{-1}, i(f, M))$.

$$\begin{aligned}
\Theta'' = S_X \cup N_X &= \{f^{-1}(\theta) \mid \theta \in \Theta' \wedge cmpd(f^{-1}(\theta), V)\} \cup N_X \\
&= \{f^{-1}(\theta) \mid \theta \in \Theta' \setminus N_Y\} \cup N_X \\
&= \{f^{-1}(f(\theta)) \mid \theta \in \Theta_s\} \cup N_X \\
&= \{\theta \mid \theta \in \Theta_s\} \cup N_X \\
&= \Theta_s \cup \{\{x_i \mapsto k, x_j \mapsto k\} \mid x_i, x_j \in X \wedge k \in D(x_i) \wedge i \neq j\} \\
&= \Theta_s \cup \Theta_{ad} \\
&= \Theta
\end{aligned}$$

S_X and N_X are the sets of no-goods in $i(f^{-1}, i(f, M))$ transformed from the explicit constraints in $i(f, M)$ and the NDA constraints respectively. Again, the ALOA constraints are discarded. All no-goods in N_Y have the form $\{y_i \mapsto k, y_j \mapsto k\}$. Applying f^{-1} to these no-goods yields invalid no-goods $\{x_k \mapsto i, x_k \mapsto j\}$. Thus, we need to consider only the no-goods in $\Theta' \setminus N_Y$, which are transformed to rediscover the set Θ_s. Furthermore, the set N_X is actually equal to Θ_{ad}. Thus, we have $\Theta'' = \Theta$. Since M and $i(f^{-1}, i(f, M))$ have the same set of no-goods, and both do not have two constraints with the same scope, we have $M = i(f^{-1}, i(f, M))$. □

Theorem 4 applies to general, possibly non-binary CSPs. (A CSP M is *binary* if $|var(c)| \leq 2$ for each constraint c in M.) Note that condition (4) in the theorem is trivially true if M is a binary Permutation CSP (a binary CSP as well as a Permutation CSP).

Corollary 5 *If the following conditions hold:*

1. *$M = ((X, D), C)$ is a binary Permutation CSP;*
2. *No two constraints in M have the same scope;*
3. *The all-different constraints in M are ensured by the no-goods $\{x_i \mapsto k, x_j \mapsto k\}$ for all $x_i, x_j \in X$ and $k \in D(x_i)$ with $i \neq j$;*
4. *We have another viewpoint connected with channeling constraints defining $f(x_i \mapsto j) = y_j \mapsto i$ for all i, j,*

then

$$M = i(f^{-1}, i(f, M)).$$

Proof 5 *All no-goods of a binary CSP must be of the form $\theta = \{x_i \mapsto a, x_j \mapsto b\}$. If $a = b$, then θ is part of the all-different constraints. Therefore, condition (4) of Theorem 4 must be satisfied. Hence the result by Theorem 4.* □

Besides, by applying model induction twice to any Permutation CSPs, condition (4) in Theorem 4 is also trivially true for the model $i(f^{-1}, i(f, M))$. Therefore, if we let $dblInduce(f, M) = i(f^{-1}, i(f, M))$ and

$$dblInduce^i(f, M) = \begin{cases} dblInduce(f, M) & \text{if } i = 1 \\ dblInduce(f, dblInduce^{i-1}(f, M)) & \text{if } i > 1, \end{cases}$$

then *dblInduce* can be idempotent. In other words, the second application of *dblInduce* results in no new redundant information.

Corollary 6 *If the following conditions hold:*

1. $M = ((X, D), C)$ *is a Permutation CSP;*
2. *No two constraints in* M *have the same scope;*
3. *The all-different constraints in* M *are ensured by the no-goods* $\{x_i \mapsto k, x_j \mapsto k\}$ *for all* $x_i, x_j \in X$ *and* $k \in D(x_i)$ *with* $i \neq j$*; and*
4. *We have another viewpoint connected with channeling constraints defining* $f(x_i \mapsto j) = y_j \mapsto i$ *for all* i, j*,*

then

$$dblInduce(f, M) = dblInduce^n(f, M) \text{ for } n \geq 1.$$

Proof 6 *After model induction from* $i(f, M)$ *to* $i(f^{-1}, i(f, M))$*, there are not any no-goods* θ *in the constraints of* $i(f^{-1}, i(f, M))$ *such that for all* $x_i, x_j \in X$ *and* $i \neq j$*,* $\pi_{\{x_i, x_j\}}(\theta) = \{x_i \mapsto k, x_j \mapsto k\}$ *except those of the all-different constraints. This is because the individual assignments of each no-good in* $i(f, M)$ *must be of different variables. Therefore, the individual assignments of each induced counterpart in* $i(f^{-1}, i(f, M))$ *must be of different values. Hence the result by Theorem 4.* □

The model *dblInduce*(f, M) differs from M in that all no-goods in M of the form $\{\ldots, x_i \mapsto k, \ldots, x_j \mapsto k, \ldots\}$ are removed. Therefore, further applying *dblInduce* to *dblInduce*(f, M) results in the same model and is not worthwhile.

4.5 Complexity Analysis

In this subsection, we focus our complexity analysis of model induction on Permutation CSPs. Given a Permutation CSP $M = ((X, D_X), C_X)$ where $X = \{x_1, \ldots, x_n\}$, $D_X(x_i) = \{1, \ldots, n\}$ for $1 \leq i \leq n$, $|C_X| = c$, and the constraints in C_X have maximum constraint arity (scope size) a. A constraint in C_X can have $O(n^a)$ no-goods. Thus, M has $O(cn^a)$ no-goods in total. Suppose we are performing model induction on M to another viewpoint (Y, D_Y), where $Y = \{y_1, \ldots, y_n\}$ and $D_Y(y_j) = \{1, \ldots, n\}$, using the channeling function $f(x_i \mapsto j) = y_j \mapsto i$ for $1 \leq i, j \leq n$. Each no-good in the constraints of M can be transformed to an induced no-good in the induced model. Furthermore, we need to include the no-goods transformed from the NDA constraints, which are $O(n^3)$ in size. Since we can skip the ALOA constraints for model induction on Permutation CSPs, the induced model has a total of $O(cn^a + n^3)$ no-goods.

In model induction, we need to apply the channeling function f to each assignment of the $O(cn^a)$ no-goods in M. By using a table lookup, f can be computed in $O(1)$ time. Since each no-good contains $O(a)$ assignments, an induced no-good can be computed in $O(a)$ time, and inducing $O(cn^a)$ no-goods takes $O(cn^a a)$ time. Furthermore, the no-goods from the NDA constraints always have a constant scope size of two. Therefore, the computation cost for $O(n^3)$ such no-goods is $O(2n^3) = O(n^3)$. Hence, the time complexity for model induction on a Permutation CSP is $O(cn^a a + n^3)$. Although the time complexity is exponential in the maximum constraint arity, handcrafting CSP models is a time-consuming task for human modelers. Therefore, human modelers should find model induction a useful tool, especially for CSPs with low constraint arities.

5 Application of Model Induction to Exploit Redundancy

While an induced model by itself is a complete specification of a problem, in this section, we focus our interest on combining induced models with other models of the same problem. We introduce two methods for combining models, namely model intersection and model channeling, and a method for merging constraints. By exploiting the applications of model intersection, channeling, and induction, we identify three new classes of combined models and show that the redundant information in the combined models allows more constraint propagation and better solving efficiency.

5.1 Combining Models

In this subsection, we describe two model combining methods, namely model intersection and model channeling. Although they are general model combining methods, we focus their applications on combining *mutually redundant* models. Model intersection and channeling allow combining two mutually redundant models in the same and different viewpoints respectively. We give their syntactic construction rules and define their set-theoretic meanings. We also discuss how the methods can help increase constraint propagation.

5.1.1 Model Intersection

Model intersection forms *conjoint models* by essentially conjoining constraints from constituent models. A solution of a conjoint model must thus also be a solution of all of its constituent models.

Given two models $M_1 = ((X_1, D_{X_1}), C_{X_1})$ and $M_2 = ((X_2, D_{X_2}), C_{X_2})$, the viewpoint $V = (X, D_X)$ of the conjoint model, denoted by $M_1 \sqcap M_2$, contains variables $X = X_1 \cup X_2$. A non-shared variable x of M_1 and M_2 has the same domain as in its constituent model, while the domain of a shared variable x is the *intersection* of $D_{X_1}(x)$ and $D_{X_2}(x)$. The constraint set C_X is the union of C_{X_1} and C_{X_2}.

Example 4 *Given two models* $M_1 = ((X_1, D_{X_1}), C_{X_1})$ *and* $M_2 = ((X_2, D_{X_2}), C_{X_2})$, *where*

- $X_1 = \{x_1, x_2, x_3\}$, $D_{X_1}(x_i) = \{1, 2, 3\}$ *for* $1 \leq i \leq 3$, *and* $C_{X_1} = \{x_1 \neq x_3\}$,
- $X_2 = \{x_2, x_3, x_4\}$, $D_{X_2}(x_i) = \{1, 2\}$ *for* $2 \leq i \leq 4$, *and* $C_{X_2} = \{x_2 = x_3\}$.

The conjoint model is $M_1 \sqcap M_2 = ((X, D_X), C_X)$, *where*

- $X = \{x_1, \ldots, x_4\}$, $D_X(x_1) = \{1, 2, 3\}$, *and* $D_X(x_i) = \{1, 2\}$ *for* $2 \leq i \leq 4$,
- $C_X = \{x_1 \neq x_3, x_2 = x_3\}$. $\square$

Note that if we represent constraints using sets of no-goods as in model induction, a no-good in a constraint of a sub-model can become invalid in the conjoint model, as the domains of the shared variables are now tightened. Therefore, such invalid no-goods should be removed from the constraints C_X of the conjoint model, i.e., $C_X = \{c' \mid c \in C_{X_1} \cup C_{X_2} \wedge var(c') = var(c) \wedge ng(c') = \{\theta \in ng(c) \mid cmpd(\theta, V)\}\}$.

A consequence of the definition of model intersection is that every solution of a conjoint model must satisfy all constraints in its constituent models.

Theorem 7 *Let V be the viewpoint of $M_1 \sqcap M_2$, then $sol(M_1 \sqcap M_2) = \{\theta_1 \cup \theta_2 \mid \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V)\}$.*

Proof 7 *Let θ_1 and θ_2 be complete assignments of M_1 and M_2 respectively.*

$$\begin{array}{ll} & \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow & \forall c_1 \in C_{X_1} \cdot [c_1\theta_1] \wedge \forall c_2 \in C_{X_2} \cdot [c_2\theta_2] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow & \forall c_1 \in C_{X_1} \cdot [c_1(\theta_1 \cup \theta_2)] \wedge \forall c_2 \in C_{X_2} \cdot [c_2(\theta_1 \cup \theta_2)] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow & \forall c \in C_X \cdot [c(\theta_1 \cup \theta_2)] \wedge cmpd(\theta_1 \cup \theta_2, V) \\ \Leftrightarrow & \theta_1 \cup \theta_2 \in sol(M_1 \sqcap M_2) \ \square \end{array}$$

Theorem 7 gives a construction of the solution set of $M_1 \sqcap M_2$ from those of M_1 and M_2. In particular, solutions θ_1 of M_1 and θ_2 of M_2 can be combined to form a solution of $M_1 \sqcap M_2$ if and only if each shared variable in θ_1 and θ_2 is assigned the same value in θ_1 and θ_2. Otherwise, $\theta_1 \cup \theta_2$ cannot form a compound assignment in the conjoint viewpoint. This condition is enforced by the *cmpd* predicate.

For the purpose of the paper, we consider mutually redundant models $M_1 = (V, C_1)$ and $M_2 = (V, C_2)$ which have the same viewpoint. Thus, $M_1 \sqcap M_2 = (V, C_1 \cup C_2)$. Furthermore, M_1, M_2, and $M_1 \sqcap M_2$ are mutually redundant to one another and have the same solution set. This property allows us to combine two models of the same problem in the same viewpoint, with the conjoint model still having the same solution set as the individual models.

Theorem 8 *Let M_1 and M_2 be mutually redundant models in the same viewpoint, then M_1, M_2, and $M_1 \sqcap M_2$ are mutually redundant to one another and $sol(M_1) = sol(M_2) = sol(M_1 \sqcap M_2)$.*

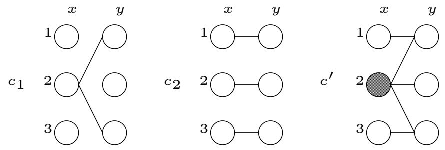


Figure 3: Increased Propagation due to Constraint Merging

Proof 8 *Since M_1 and M_2 are mutually redundant and in the same viewpoint, the identity mapping is the only one-to-one mapping between $sol(M_1)$ and $sol(M_2)$, i.e., $sol(M_1) = sol(M_2)$. By Theorem 7, $sol(M_1 \sqcap M_2) = sol(M_1) = sol(M_2)$ and the three models are mutually redundant to one another.* □

Model intersection is "additive" in nature; it collects constraints from both constituent models to increase the source of constraint propagation. For example, suppose M_1 and M_2 are in the same viewpoint and $x+y < 3$ and $x-y > 1$ are two constraints in M_1 and M_2 respectively, with $D(x) = D(y) = \{0, \ldots, 4\}$. Constraint propagation in M_1 removes the values 3 and 4 from $D(x)$ and $D(y)$. Similarly, propagation in M_2 removes 0 and 1 from $D(x)$ and 3 and 4 from $D(y)$. In the conjoint model $M_1 \sqcap M_2$, the information of domain reduction is combined, causing x be automatically assigned the value 2. The assignment $x \mapsto 2$ makes all $j \in D(y)$ with $j < 1$ be removed, since both $2 + y < 3$ and $2 - y > 1$ imply $y < 1$. Hence, $D(y) = \{0\}$, i.e., y is bound to value 0, and the assignment $y \mapsto 0$ can trigger further propagation.

When using sets of no-goods to represent constraints, we have the option of *merging constraints* with the same scope into one constraint by taking the union of the constraints' sets of no-goods. Given two constraints c_1 and c_2 with $var(c_1) = var(c_2)$, we can construct a merged constraint c' to replace c_1 and c_2 such that $var(c') = var(c_1) = var(c_2)$ and $ng(c') = ng(c_1) \cup ng(c_2)$. The resultant constraint c', logically equivalent to $c_1 \wedge c_2$, potentially provides more constraint propagation than the individual constraints c_1 and c_2 do when used separately. For example, suppose $D(x) = D(y) = \{1, 2, 3\}$ and consider two constraints $c_1 : x = 2 \Rightarrow y = 2$ and $c_2 : x \neq y$. Both c_1 and c_2 have the same scope $var(c_1) = var(c_2) = \{x, y\}$. Figure 3 shows the configurations of the two constraints and the merged constraint c'. Although both c_1 and c_2 are arc consistent individually, the merged constraint c' is not; the value 2 in $D(x)$ has no support in $D(y)$ and hence can be removed from $D(x)$. Constraint merging is therefore beneficial. It is applicable whenever there is more than one constraint having the same scope in a CSP. Furthermore, it arises more naturally from model intersection, since it is quite plausible for two constituent models to have different constraints with the same scope.

5.1.2 Model Channeling

Suppose there is a set C_c of channeling constraints connecting the viewpoints V_1 and V_2. *Model channeling* [5] combines $M_1 = ((X_1, D_{X_1}), C_{X_1})$ and $M_2 =$

$((X_2, D_{X_2}), C_{X_2})$ using C_c to form a *channeled model*, denoted by $M_1 \overset{C_c}{\bowtie} M_2$, which is $M_1 \sqcap M_2$ plus the channeling constraints C_c, i.e., the constraints in $M_1 \overset{C_c}{\bowtie} M_2$ is $C_{X_1} \cup C_{X_2} \cup C_c$.

Given two models M_1 and M_2, the channeled model $M_1 \overset{C_c}{\bowtie} M_2$ is more constrained than the conjoint model $M_1 \sqcap M_2$. A solution of $M_1 \overset{C_c}{\bowtie} M_2$ must be a solution of $M_1 \sqcap M_2$ and also satisfy the channeling constraints C_c.

Theorem 9 *Let V be the viewpoint of $M_1 \overset{C_c}{\bowtie} M_2$, then $sol(M_1 \overset{C_c}{\bowtie} M_2) = \{\theta_1 \cup \theta_2 \mid \theta_1 \in sol(M_1) \wedge \theta_2 \in sol(M_2) \wedge cmpd(\theta_1 \cup \theta_2, V) \wedge \forall c \in C_c \cdot c(\theta_1 \cup \theta_2)\}$.*

Proof 9 *Similar to the proof of Theorem 7.* □

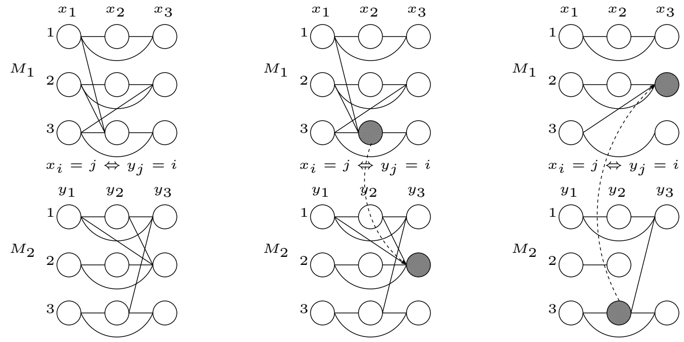
Redundant modeling [5] is an application of model channeling to two mutually redundant models.

Theorem 10 *Suppose M_1 and M_2 are mutually redundant models. If C_c is a set of channeling constraints enforcing the one-to-one mapping between $sol(M_1)$ and $sol(M_2)$, then M_1, M_2, and $M_1 \overset{C_c}{\bowtie} M_2$ are mutually redundant to one another.*

Proof 10 *By definition, any solution θ of $M_1 \overset{C_c}{\bowtie} M_2$ can be projected on V_1 and V_2 to form solutions of M_1 and M_2 respectively. Suppose θ_1 is a solution of M_1. Then there must be θ_2 that can be mapped to θ_1 by C_c (and vice versa) and is a solution of M_2. In addition, $cmpd(\theta_1 \cup \theta_2, V)$, where V is the viewpoint of $M = M_1 \overset{C_c}{\bowtie} M_2$, must hold since $\theta_1 \cup \theta_2$ satisfies C_c and shared variables in θ_1 and θ_2 must share the same assignments. Therefore, there is a one-to-one mapping between $sol(M_1)$ and $sol(M)$. Similarly, there is also a one-to-one mapping between $sol(M_2)$ and $sol(M)$.* □

Model channeling is "collaborative" in the sense that it allows each sub-model to perform constraint propagation on its own, and yet communicate its results (variable instantiation and domain pruning) to the other sub-model via the channeling constraints to possibly initiate further constraint propagation. Furthermore, model channeling allows constraint propagation to explore different variable spaces (viewpoints). For example, Figure 4(a) shows two mutually redundant models M_1 and M_2 being channeled with the channeling constraints C_c $x_i = j \Leftrightarrow y_j = i$ for all i, j pairs. M_1 has three constraints c_{12}, c_{13}, and c_{23}, each contains the no-goods:

$$\begin{array}{ll}
c_{12}: & \{x_1 \mapsto 1, x_2 \mapsto 1\}, \{x_1 \mapsto 1, x_2 \mapsto 3\}, \{x_1 \mapsto 2, x_2 \mapsto 2\}, \\
 & \{x_1 \mapsto 2, x_2 \mapsto 3\}, \{x_1 \mapsto 3, x_2 \mapsto 3\}, \\
c_{13}: & \{x_1 \mapsto 1, x_3 \mapsto 1\}, \{x_1 \mapsto 2, x_3 \mapsto 2\}, \{x_1 \mapsto 3, x_3 \mapsto 2\}, \{x_1 \mapsto 3, x_3 \mapsto 3\}, \\
c_{23}: & \{x_2 \mapsto 1, x_3 \mapsto 1\}, \{x_2 \mapsto 2, x_3 \mapsto 2\}, \{x_2 \mapsto 3, x_3 \mapsto 3\}.
\end{array}$$



(a) Initial State (b) Intermediate State (c) Final State

Figure 4: Constraint Propagation on Model Channeling between Two Models

M_2 has three constraints c'_{12}, c'_{13}, and c'_{23}, each contains the no-goods

$$
\begin{array}{rl}
c'_{12}: & \{y_1 \mapsto 1, y_2 \mapsto 1\}, \{y_1 \mapsto 2, y_2 \mapsto 2\}, \{y_1 \mapsto 3, y_2 \mapsto 3\}, \\
c'_{13}: & \{y_1 \mapsto 1, y_3 \mapsto 1\}, \{y_1 \mapsto 1, y_3 \mapsto 2\}, \{y_1 \mapsto 2, y_3 \mapsto 2\}, \{y_1 \mapsto 3, y_3 \mapsto 3\}, \\
c'_{23}: & \{y_2 \mapsto 1, y_3 \mapsto 1\}, \{y_2 \mapsto 1, y_3 \mapsto 2\}, \{y_2 \mapsto 2, y_3 \mapsto 2\}, \\
& \{y_2 \mapsto 3, y_3 \mapsto 1\}, \{y_2 \mapsto 3, y_3 \mapsto 3\}.
\end{array}
$$

Constraint propagation in M_1 alone removes value 3 from $D_X(x_2)$ but no more, as $x_2 \mapsto 3$ is incompatible to all assignments of x_1. Propagation in M_2 alone removes no values. In the channeled model $M_1 \overset{C_c}{\bowtie} M_2$, however, the removal of $x_2 \mapsto 3$ causes $y_3 \mapsto 2$ to be removed also, as shown in Figure 4(b). Note that $y_3 \mapsto 2$ is the only support in variable y_3 for the assignment $y_2 \mapsto 3$. Thus, removing $y_3 \mapsto 2$ makes $y_2 \mapsto 3$ incompatible with all the remaining assignments of y_3. As a result, $y_2 \mapsto 3$ in M_2 and consequently $x_3 \mapsto 2$ in M_1 are removed also. Figure 4(c) shows the final state of propagation, showing that the channeled model allows more constraint propagation than either M_1 or M_2 alone does.

5.2 Three New Classes of Combined Models

A viewpoint can greatly influence how a human modeler looks at a problem. Each viewpoint provides a distinct perspective emphasizing perhaps a specific aspect of the problem. Therefore, if a modeler handcrafts two CSP models of the same problem in two different viewpoints, the constraints in the two models are likely to be expressed very differently. A constraint expressed in one viewpoint might not even have an (explicit) counterpart in the other [30]. However, since the CSP models are modeling the same problem, the constraints in each model should give the same solutions to the problem being modeled.

Suppose $M_1 = (V_1, C_1)$ and $M_2 = (V_2, C_2)$ are mutually redundant models in different viewpoints handcrafted by human modeler, and C_c is a set of channeling constraints which defines a total function f from possible assignments of

V_1 to those of V_2. Model channeling and intersection give various possibilities to combine M_1, M_2, and their induced models. Assuming f^{-1} *exists*, we propose three classes of interesting combined models.

1. $i(f, M_1) \sqcap M_2$ and $M_1 \sqcap i(f^{-1}, M_2)$
2. $M_1 \overset{C_c}{\bowtie} i(f, M_1)$ and $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$
3. $(i(f, M_1) \sqcap M_2) \overset{C_c}{\bowtie} i(f^{-1}, i(f, M_1) \sqcap M_2)$ and $(i(f^{-1}, M_2) \sqcap M_1) \overset{C_c}{\bowtie} i(f, i(f^{-1}, M_2) \sqcap M_1)$

We first exploit the use of model intersection to combine two mutually redundant models in the same viewpoint. Model induction of M_1 essentially translates constraints expressed in V_1 to constraints in V_2 via the channeling function f. The transformed constraints express in V_2 the constraint information of the problem as viewed from V_1; they are likely to be different from constraints expressed directly using V_2 by a human modeler. Therefore, $i(f, M_1)$ and M_2, which have the same viewpoint V_2, are mutually redundant and yet complementary to each other. In the first class, we conjoin the two models to form $M_{1^i2} = i(f, M_1) \sqcap M_2$. Similarly, we can construct the model $M_{12^i} = M_1 \sqcap i(f^{-1}, M_2)$.

A model M_1 is mutually redundant with its induced model $i(f, M_1)$ according to Theorem 1. As the two models are in different viewpoints V_1 and V_2 respectively, in the second class, we use model channeling to combine them using the channeling constraints C_c, giving the channeled model $M_1 \overset{C_c}{\bowtie} i(f, M_1)$. Such application of model channeling allows us to enjoy the benefits of combining models even when we have only one handcrafted model. The other model that we can construct in this class is $M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$.

In the first and second classes, we make use of model intersection and model channeling respectively to form combined models. In the third class, we apply both methods simultaneously to form further combined models. We first conjoin the models $i(f, M_1)$ and M_2 as suggested in the first class. Since the conjoint model M_{1^i2} is in a single viewpoint V_2, we can then apply model induction to M_{1^i2} using f^{-1}, giving the induced model $i(f^{-1}, M_{1^i2})$. By Theorem 1, M_{1^i2} and $i(f^{-1}, M_{1^i2})$ are mutually redundant. Using the technique in the second class, we channel the two models using channeling constraints C_c, giving the combined model $M_{1^i2} \overset{C_c}{\bowtie} i(f^{-1}, M_{1^i2})$. The other model in the third class is therefore $M_{12^i} \overset{C_c}{\bowtie} i(f, M_{12^i})$.

Note that f^{-1} always exists for Permutation CSPs. Therefore, we can always perform model induction using either M_1 or M_2. The model M_{1^i2} in the first class combines the constraint information of both handcrafted models M_1 and M_2 in the single viewpoint V_2. By Corollary 6, we cannot construct further induced models that are different from the result of the second induction, i.e., we need not consider the models $dblInduce^k(f, M_{1^i2})$ for $k > 1$. Furthermore, $dblInduce(f, M_{1^i2})$ contains fewer no-goods than M. Hence, in viewpoint V_2,

M_{1^i2} is the model that has the largest number of no-goods. The model $M_{1^i2} \overset{C_c}{\bowtie} i(f^{-1}, M_{1^i2})$ in the third class, in the combined viewpoint of V_1 and V_2, has therefore exploited the most redundant information from the original models M_1 and M_2 using model induction, intersection, and channeling.

Also note that for the special case of *binary* Permutation CSPs, $i(f^{-1}, i(f, M_1) \sqcap M_2) = M_1 \sqcap i(f^{-1}, M_2)$ with constraint merging. Similarly, $i(f, i(f^{-1}, M_2) \sqcap M_1) = M_2 \sqcap i(f, M_1)$. Thus, the second model in the third class $(i(f^{-1}, M_2) \sqcap M_1) \overset{C_c}{\bowtie} i(f, i(f^{-1}, M_2) \sqcap M_1)$ becomes $i(f^{-1}, i(f, M_1) \sqcap M_2) \overset{C_c}{\bowtie} (M_2 \sqcap i(f, M_1))$, which is actually the first model in the class. Therefore, it is only necessary to consider one of the two models for the third class in the case of binary Permutation CSP with constraint merging.

Corollary 11 *If the following conditions hold:*

- $M_1 = ((X, D_X), C_X)$ *and* $M_2 = ((Y, D_Y), C_Y)$ *are mutually redundant Permutation CSPs and* M_1 *is binary;*
- *The all-different constraints in* M_1 *are ensured by the no-goods* $\{x_i \mapsto k, x_j \mapsto k\}$ *for all* $x_i, x_j \in X$ *and* $k \in D_X(x_i)$ *with* $i \neq j$*; and*
- (X, D_X) *and* (Y, D_Y) *are connected with channeling constraints defining* $f(x_i \mapsto j) = y_j \mapsto i$ *for all* i, j*,*

then

$$i(f^{-1}, i(f, M_1) \sqcap M_2) = M_1 \sqcap i(f^{-1}, M_2) \textit{ with constraint merging.}$$

Proof 11 *With constraint merging, we have*

$$i(f^{-1}, i(f, M_1) \sqcap M_2) = i(f^{-1}, i(f, M_1)) \sqcap i(f^{-1}, M_2).$$

By Theorem 4, $i(f^{-1}, i(f, M_1)) = M_1$*. Hence,*

$$i(f^{-1}, i(f, M_1) \sqcap M_2) = i(f^{-1}, i(f, M_1)) \sqcap i(f^{-1}, M_2) = M_1 \sqcap i(f^{-1}, M_2) \ \Box$$

6 Experiments

In this section, we present experimental results on combining mutually redundant models using our proposed scheme. The experimental problems are Langford's problem, which can be modeled as Permutation CSPs, and random Permutation CSPs. We realize and evaluate various models of these problems using ILOG Solver 4.4 [1], a tree search based CSP solver, running on a Sun Blade 2500 (2 x 1.6GHz US-IIIi) workstation with 2GB memory. Note that our experiments aim to show that using a standard level of constraint propagation in Solver, the redundant information obtained by model induction, intersection, and channeling can help reduce search effort in practice. It is outside the scope of this paper to consider (1) the effects of maintaining different levels of constraint propagation in the combined models and (2) whether each piece of redundant information in the combined models is propagation redundant [9, 10] with respect to some other constraints.

We implement the REVISE2001/3.1 procedure [4] to enforce arc consistency on constraints represented extensionally by sets of no-goods. Since we explicitly use no-goods to build constraints, it is straightforward to apply constraint merging. Hence, we merge constraints whenever possible so that every constraint in a model has a unique scope. The channeling constraints are expressed using Solver's *inverse* constraint [1], which is essentially equivalent to the set of constraints $x_i = j \Leftrightarrow y_j = i$ for all i, j pairs, although more efficiently implemented. The smallest-domain-first dynamic variable ordering heuristic is used throughout the experiments.

6.1 Langford's Problem

Langford's problem[2] can be modeled as Permutation CSPs for experimenting with model induction, intersection, and channeling. The problem is to find a $(m \times n)$-number sequence which includes the numbers 1 to n, with each number occurring m times, such that two consecutive occurrences of number i are separated by i positions. An instance is denoted by (m, n). A solution of the $(2, 3)$ instance is the sequence $\langle 2, 3, 1, 2, 1, 3 \rangle$.

Smith [34] suggested two Permutation CSP models for Langford's problem. We use the $(2, 3)$ instance to illustrate them. In the first model M_1, we use six variables $X = \{x_0, \ldots, x_5\}$, which we can think of as $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2\}$. Here, i_j represents the j-th occurrence of number i in a sequence. The variable domain is the possible positions of a number in the sequence. We use $\{0, \ldots, 5\}$ to represent the domain, giving us the viewpoint $V_1 = (X, D_X)$ where $D_X(x_i) = \{0, \ldots, 5\}$ for $0 \le i \le 5$.

Using V_1, we can model the problem using two types of constraints. The all-different constraints ensure that all numbers occur in different positions in the sequence, whereas the separation constraints ensure the correct spacings between consecutive occurrences of the same number.

all-different constraints: $x_i \neq x_j$ for $0 \le i < j \le 5$

separation constraints: $x_1 = x_0 + 2$, $x_3 = x_2 + 3$, and $x_5 = x_4 + 4$

In the second model M_2, we again use 6 variables $Y = \{y_0, \ldots, y_5\}$ to represent each position in the sequence. Their domains are $\{0, \ldots, 5\}$, corresponding to the numbers $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2\}$. Hence, we have the viewpoint $V_2 = (Y, D_Y)$ where $D_Y(y_j) = \{0, \ldots, 5\}$ for $0 \le j \le 5$.

Using this viewpoint, we also have two types of constraints as in V_1:

all-different constraints: $y_i \neq y_j$ for $0 \le i < j \le 5$

[2] "Prob024" in CSPLib, available at `<http://www.csplib.org/>`.

separation constraints: $y_j = 0 \Leftrightarrow y_{j+2} = 1$ for $0 \leq j \leq 3$, $y_j = 2 \Leftrightarrow y_{j+3} = 3$ for $0 \leq j \leq 2$, $y_j = 4 \Leftrightarrow y_{j+4} = 5$ for $0 \leq j \leq 1$, $y_j \neq 0$ for $4 \leq j \leq 5$, $y_j \neq 2$ for $3 \leq j \leq 5$, and $y_j \neq 4$ for $2 \leq j \leq 5$

We can express the channeling constraints C_c connecting V_1 and V_2 as $x_i = j \Leftrightarrow y_j = i$ for $0 \leq i, j \leq 5$. These constraints define a total and bijective function f where $f(x_i \mapsto j) = y_j \mapsto i$ for all valid i, j. With M_1, M_2, and f, we can construct the three proposed classes of combined models. Since both M_1 and M_2 are binary Permutation CSPs, Corollary 11 suggests that it is only necessary to consider

$$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2} = (M_1 \sqcap i(f^{-1}, M_2)) \overset{C_c}{\bowtie} (i(f, M_1) \sqcap M_2)$$

for the third class of combined models.

Note that the separation constraints are expressed differently in the two viewpoints. For example, to express in the $(2, 3)$ instance that 1_1 and 1_2 are separated by one position, we use simply one constraint $x_1 = x_0 + 2$ in M_1, which is shown in Figure 5(a). Figure 5(b) shows its counterpart in $i(f, M_1)$. The same requirement in M_2, however, has to be expressed by six constraints $y_0 = 0 \Leftrightarrow y_2 = 1$, $y_1 = 0 \Leftrightarrow y_3 = 1$, $y_2 = 0 \Leftrightarrow y_4 = 1$, $y_3 = 0 \Leftrightarrow y_5 = 1$, $y_4 \neq 0$, and $y_5 \neq 0$, as shown in Figure 5(c). Although $i(f, M_1)$ and M_2 have the same viewpoint V_2, the same requirement is represented differently. Therefore, combining these mutually redundant models using model intersection can enhance constraint propagation. Figure 5(d) shows the constraints in the conjoint model $i(f, M_1) \sqcap M_2$.

We evaluate the various models for some instances of Langford's problem. Tables 1 and 2 show our comparison results of the satisfiable and unsatisfiable instances respectively. Column 1 gives the models. In models with more than one viewpoint, it suffices to assign values to the variables of either viewpoint, or one can choose to search on both sets of variables: in column 2, we give the search variables used. The remaining columns report the execution results. We report the number of fails (i.e., the number of backtracks in a search tree, the smaller the better) and CPU time in seconds of an execution. A cell labeled with "-" means that execution does not terminate within two hours of CPU time. In Table 1, we report the results for finding the first solution, while in Table 2, we report the results for proving unsatisfiability of the instances. We also highlight in bold the best result of each column.

We divide the models into five groups. The first group consists of individual models, while the second group consists of combined models constructed using the redundant modeling approach [5]. The remaining groups correspond to our three proposed classes of combined models.

In analyzing the results, attention should be directed not just to the CPU time, but also to the number of fails (i.e., the number of times the search backtracks). In fact, the latter is more important and accurate as a measure of the robustness of a model. Combined models are bigger in size, and higher execution overhead is expected. The idea of combining mutually redundant

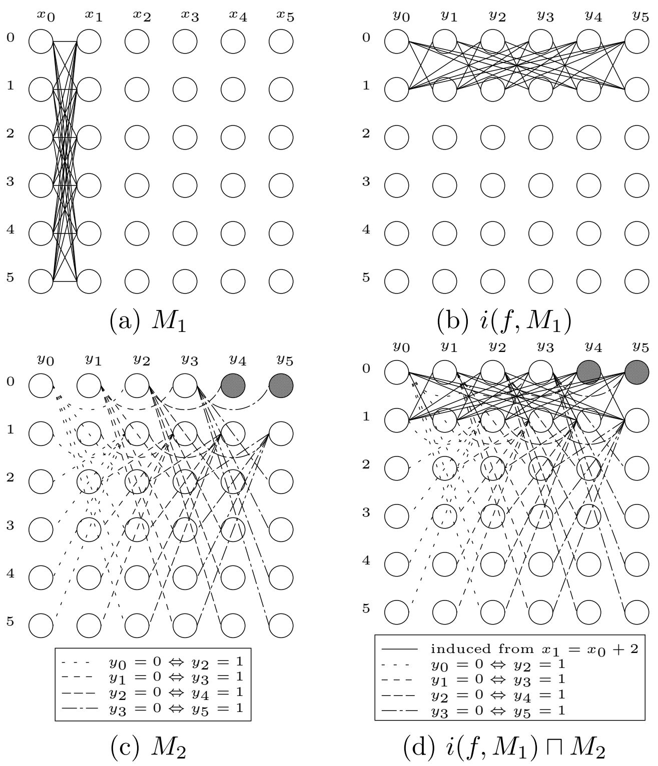


Figure 5: The Constraints for Correct Separation of 1_1 and 1_2 in M_1, $i(f, M_1)$, M_2 and $i(f, M_1) \sqcap M_2$ of the $(2, 3)$ Instance Respectively

models is to spend more time in constraint propagation at each node of the search tree, in the hope that the extra effort will result in more substantial pruning of the search space. A model giving more pruning has a greater chance of solving problems that are otherwise computationally infeasible than those with less pruning.

The first and fifth groups of models represent the two ends of a spectrum, which indicates the amount of model redundancy utilized in the models. The single models in the first group use no redundancy, and thus perform the worst in terms of the number of fails. Their execution times are not the worst since these models are the smallest in size, incurring the least execution overhead in constraint propagation. Model M_2 is a poor model. Any model involving M_2 as a base model performs poorly, in terms of both CPU time and number of fails. In the following, we focus on only models using M_1 as a base model.

The second group makes use of only model channeling, which helps M_1 and M_2 share pruning and variable instantiation information. Constraint propagation also takes place in both viewpoints. Another advantage of this approach is that constraints in M_1 and M_2, constructed under different viewpoints, are

Table 1: Comparison Results of Finding First Solution of Langford's Problem

Models	Search Variables	$(3,9)$ Fails	Time	$(3,10)$ Fails	Time	$(3,17)$ Fails	Time	$(3,18)$ Fails	Time	$(3,19)$ Fails	Time
M_1	X	192	0.27	569	1.01	11242	82.54	34134	304.71	132874	1222.18
$i(f, M_1)$	Y	-	-	-	-	-	-	-	-	-	-
M_2	Y	-	-	-	-	-	-	-	-	-	-
$i(f^{-1}, M_2)$	X	-	-	-	-	-	-	-	-	-	-
$M_1 \overset{C_c}{\bowtie} M_2$	X	63	0.28	193	0.88	4251	61.15	11491	203.63	47004	805.62
$M_1 \overset{C_c}{\bowtie} M_2$	Y	44	0.2	20	**0.19**	2967	34.13	10314	144.97	13557	194.3
$M_1 \overset{C_c}{\bowtie} M_2$	$X \cup Y$	39	0.22	104	0.63	140	3.16	2004	33.07	1729	33.23
$M_1 \sqcap i(f^{-1}, M_2)$	X	105	**0.19**	313	0.62	6054	44.6	18174	161.39	72295	648.69
$i(f, M_1) \sqcap M_2$	Y	-	-	-	-	-	-	-	-	-	-
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	X	73	0.32	207	1.02	4164	69.37	11227	205.16	46285	896.17
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	Y	47	0.22	21	0.22	2928	41.18	9997	179.52	13161	220.96
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	$X \cup Y$	42	0.26	113	0.81	152	3.45	2031	43.77	1834	37.64
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	X	115	0.58	340	2.2	6031	135.63	18103	491.12	70777	1908.38
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	Y	589	2.09	475	2.22	14182	216.08	55205	931.03	178472	3379.64
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	$X \cup Y$	113	0.57	338	2.22	5808	132.58	17102	463.66	67024	1898.69
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	X	38	0.26	132	0.77	2750	45.67	7327	155.12	29844	627.53
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	Y	31	**0.19**	**9**	0.2	1704	22.68	6116	90.59	6856	119.91
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	$X \cup Y$	**29**	0.24	83	0.62	**113**	**2.97**	**1468**	**27.87**	**1281**	**28.42**

Table 2: Comparison Results of Proving Unsatisfiability of Langford's Problem

Models	Search Variables	$(3,11)$ Fails	Time	$(3,12)$ Fails	Time	$(3,13)$ Fails	Time	$(4,11)$ Fails	Time	$(4,12)$ Fails	Time	$(4,13)$ Fails	Time
M_1	X	14512	30.86	62016	170.38	300800	956.07	1876	15.97	6284	61.76	23474	305.49
$i(f, M_1)$	Y	-	-	-	-	-	-	-	-	-	-	-	-
M_2	Y	-	-	-	-	-	-	-	-	-	-	-	-
$i(f^{-1}, M_2)$	X	-	-	-	-	-	-	-	-	-	-	-	-
$M_1 \overset{C_c}{\bowtie} M_2$	X	4353	21.99	18057	105.35	80828	650.2	600	11.78	1691	36.11	5465	157.94
$M_1 \overset{C_c}{\bowtie} M_2$	Y	3657	17.87	14058	82.91	55312	392.46	384	8.41	1096	25.65	2688	84.3
$M_1 \overset{C_c}{\bowtie} M_2$	$X \cup Y$	2692	15.35	10411	72.32	40475	342.57	306	7.95	881	24	2070	67.18
$M_1 \sqcap i(f^{-1}, M_2)$	X	7621	17.11	33254	91.05	160855	543.7	1078	8.61	3251	30.63	12585	136.77
$i(f, M_1) \sqcap M_2$	Y	-	-	-	-	-	-	-	-	-	-	-	-
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	X	4873	27.52	19587	135.34	87381	725.68	609	13.52	1736	50.14	5631	179.68
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	Y	3801	22.43	14604	103.18	57548	471.72	403	9.81	1108	32.68	2719	100.83
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	$X \cup Y$	2774	17.83	11042	86.08	42748	396.17	317	8.82	894	28.93	2059	80.98
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	X	8130	58.72	35246	322.92	168438	1741.31	1160	30.86	3647	122.76	13486	505.69
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	Y	25344	140.85	108234	794.67	530309	4382.1	7918	120.97	24800	454.96	97091	2136.8
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	$X \cup Y$	7907	58.2	34195	290.05	161057	1730.08	1155	30.53	3636	114.54	13263	496.29
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	X	2973	18.32	12739	87.35	55570	477.08	392	9.55	1307	30.21	3828	110.63
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	Y	2553	15.43	9659	67.68	37887	323.73	287	**6.42**	799	21.98	1988	64.96
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	$X \cup Y$	**1937**	**13.14**	**7623**	**59.66**	**29662**	**286.27**	**268**	6.59	**692**	**20.2**	**1630**	**55.1**

complementary to each other. These characteristics are the source of increased constraint propagation; and thus the drastic cut in the number of fails as compared to the models in the first group.

The third group of models uses only one viewpoint, but model intersection and constraint merging combine the constraints from the two models to form stronger constraints to achieve stronger constraint propagation. The reduction in the number of fails, however, is not as substantial as the case in the second group of models.

The fourth group of models employs both model induction and model channeling. The models inherit the good characteristics of model channeling, except that the constraints are based only on *one* original model, and so do not have the chance to share constraint information from the other. Therefore, the performance of the fourth group is slightly worse than that of the second group.

The model in the fifth group enjoys the best of both worlds. Each of the sub-models is a conjoint model, which encompasses strengthened constraints obtained from model intersection. The conjoint models are then connected via model channeling to take advantage of the sharing of pruning information and constraint propagation in different viewpoints. That explains why models in this group always give the lowest number of fails. Their timings are also the fastest in most instances, although these models are the largest in size.

6.2 Random Permutation CSPs

A random (binary) CSP can be characterized by (n, m, p_1, p_2), where n is the number of variables and m is the domain size of the variables. The constraint density p_1 is the proportion of pairs of variables that have a constraint between them. Given a constraint between a variable pair, the constraint tightness p_2 is the proportion of value pairs that are inconsistent for the variable pair.

In a Permutation CSP, the number of variables always equals the domain size, i.e., $n = m$. Thus, a random Permutation CSP can be characterized by (n, p_1, p_2). It is generated by artificially adding to a random CSP (n, n, p_1, p_2) the no-goods $\{x_i \mapsto k, x_j \mapsto k\}$ for all i, j, k pairs with $i \neq j$, which represent the all-different constraints of the random CSP. Therefore, the actual constraint density in a random Permutation CSP is 1.0 due to the all-different constraints, but only a proportion p_1 of them can have no-goods other than those of the all-different constraints. Note that given a constraint between two variables in a random Permutation CSP, the expected number of no-goods in the constraint is $(n^2 - n)p_2 + n$ instead of $n^2 p_2$ in a constraint of a random CSP.

For a Permutation CSP, a second viewpoint always exists by interchanging the roles of variables and values. The channeling constraints connecting the two viewpoints are the same as those in Langford's problem, i.e., $x_i = j \Leftrightarrow y_j = i$, giving the channeling function $f(x_i \mapsto j) = y_j \mapsto i$ for all i, j pairs. Hence, we can always construct an induced model $i(f, M)$ for a Permutation CSP M.

Our experiments on random Permutation CSPs are divided in two categories. In the first set of experiments, we use single random models to study the effect of the induced models with model channeling. In the second, we use two mutually redundant random models to experiment with both model intersection and channeling.

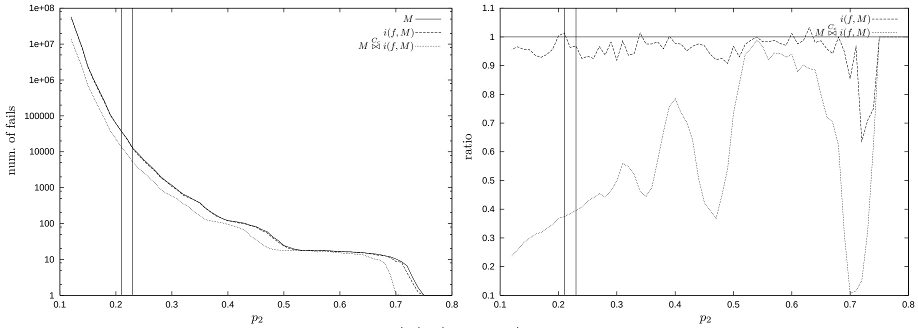
6.2.1 Single Models

We evaluate the models M, $i(f, M)$, and $M \overset{C_c}{\bowtie} i(f, M)$ for the instances of the random Permutation CSPs. We fix $n = m = 18$ and $p_1 \in \{1.0, 0.8, 0.6\}$, and vary p_2 in steps of 0.01 up to 1.0. The series of experiments with a particular n and p_1 is denoted by (n, p_1). We generate ten instances for each set of parameters and plot the average number of fails and average CPU time in the graphs.

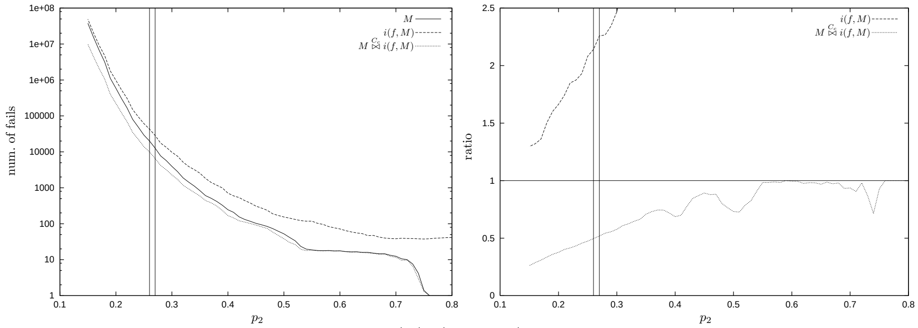
The left-hand-side graphs in Figs. 6(a), 6(b), and 6(c) show the amount of search in terms of number of fails for finding *all* solutions for the series $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$ at different p_2 levels respectively. In $M \overset{C_c}{\bowtie} i(f, M)$, for graph clarity, we plot only the curves of searching the variables of both constituent models, since this searching strategy generally performs better than searching the variables in either model. The right-hand-side graphs in the figures show the ratio of the number of fails of $i(f, M)$ or $M \overset{C_c}{\bowtie} i(f, M)$ to that of M. A model with ratio higher (lower) than 1.0 means it needs more (less) search than the original model does. Hence, a lower ratio is better. A horizontal line is drawn at ratio 1.0 to distinguish the two regions. The vertical lines in the graphs indicate the *mushy region* [36] in which phase transition occurs, i.e., some of the instances are satisfiable but some are unsatisfiable.

Figs. 6(a), 6(b), and 6(c) show that the number of fails decreases smoothly as p_2 increases. This is in contrary to the case of first solution search, where a peak is found during phase transition. At high constraint tightness, constraint propagation alone is enough to prove the instances unsatisfiable. Hence, the number of fails is 1. The ratios generally increase with p_2 regardless of the mushy regions and gradually converge to 1.0 because at high p_2 values, both the original models and channeled models need no search to prove unsatisfiability and the number of fails (and hence the ratio) is 1 at that case. Note that the ratios when $p_1 = 1.0$ are generally lower than those at the other p_1 values. Also, in the mushy regions, the ratio decreases with increasing p_1. For example, the average ratios in the mushy regions of $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$ are 0.38, 0.51, and 0.63 respectively. These indicate that model channeling is useful to reduce search space in CSPs with variables highly connected by constraints.

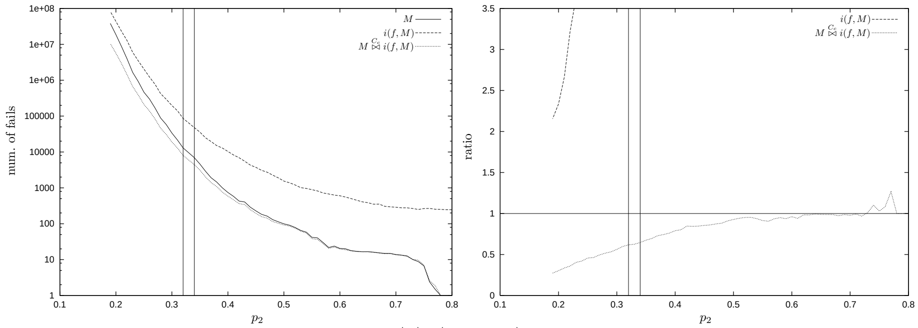
We can also see from the figures that an induced model alone performs no better than the original model. Only at $p_1 = 1.0$ are the induced models competitive. At lower p_1 values, they perform worse than the original models, since a low p_1 in M means a low p_2 in the induced model $i(f, M)$, which implies that a domain value of a variable is less likely to be inconsistent. In particular, even at high p_2 levels, we still need tree search to prove the instances unsatisfiable. Figure 7 shows an extreme example of such situation. In the original model on the left of the figure, the constraint between x_1 and x_2 can never be satisfied as it contains all the possible no-goods. Before tree search, constraint propagation alone will prune all the domain values of x_1 or x_2 to prove the model unsatisfiable. The induced model on the right, however, is clearly still arc consistent, since domain values 1 and 2 of a variable are always consistent with value 3 in another variable, and vice versa. Hence, constraint propagation prunes no do-



(a) $(18, 1.0)$



(b) $(18, 0.8)$



(c) $(18, 0.6)$

Figure 6: Average Number of Fails and Ratios of Fails to the Single Models for the Series $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$ Respectively

main values and tree search is needed to prove the induced model unsatisfiable. Only when $p_1 = 1.0$ can we detect arc inconsistencies in the induced model before tree search. This explains why the induced models alone are difficult to solve when $p_1 < 1.0$. Nevertheless, they are useful to combine with other models through intersection or channeling to obtain extra redundant information.

Figs. 8(a), 8(b), and 8(c) show the timing results of the series $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$ respectively. Despite the drastic pruning of the search spaces, the channeled models do not always have faster run times than the

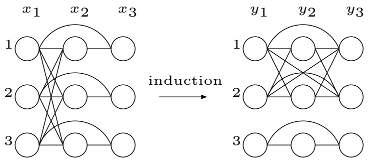


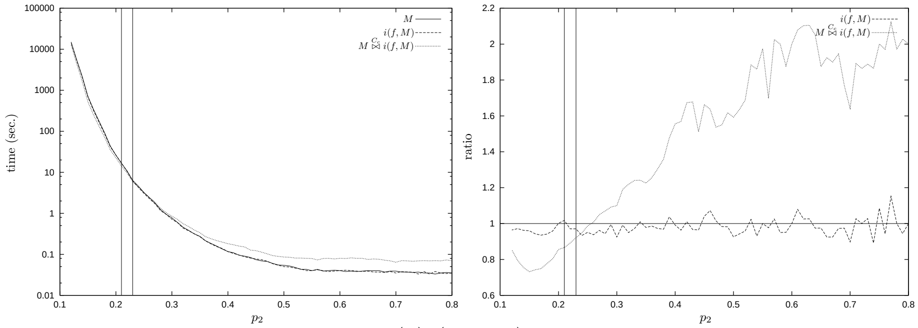
Figure 7: An Unsatisfiable CSP and its Induced Model

single models, as they are larger in size and constraint propagation takes longer at each node of the search tree. When p_1 is high, the run times of the channeled models are usually better than those of the single models in or before the mushy region. For example, for $p_1 = 1.0$, a combined model is more efficient than a single model when $p_2 \leq 0.25$. As p_2 increases to cross the region, the instances become easier to prove unsatisfiability. The longer time spent on propagating the channeled models cannot be compensated and the ratios reach 2.0 on high p_2 values. Note that in our experiments, there are cases that a single model requires much longer run time than its corresponding channeled model, and the ratio is more than an order of magnitude. This shows that while a channeled model in general can have propagation overhead over a single model, it is possible that in some circumstances, the channeled model is substantially more efficient than the single model and can tackle a problem which is computationally infeasible for the single model. Besides, since the induced models alone for $p_1 < 1.0$ requires more search than the single models, they have a comparatively longer run time, especially at high p_2 values.

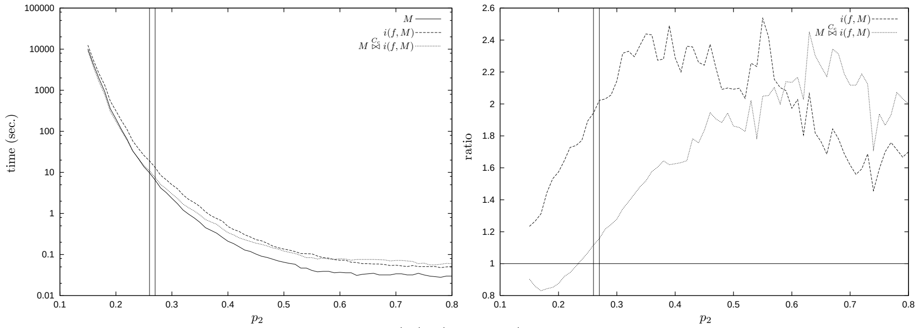
6.2.2 Mutually Redundant Random Model Pairs

In order to apply model intersection, we need, as in Langford's problem, two mutually redundant models in the same viewpoint. For Random Permutation CSPs, we do not have two such models in general, since the solutions of two random instances are usually not equivalent. Therefore, we have designed an algorithm [28] to generate pairs of mutually redundant Random Permutation CSPs M_1 and M_2 of the same class, by forcing them to have equivalent sets of solutions with respect to the channeling constraints. Table 3 shows the results of finding all solutions for the instances in the mushy regions of $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$. The reported figures are the average results of the satisfiable instances in a class.

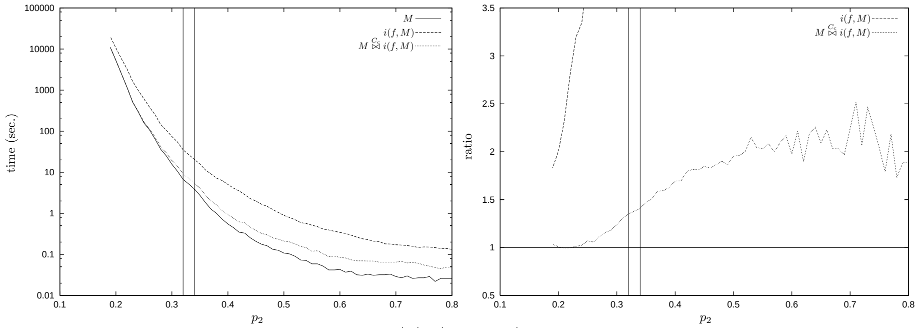
Since the generated models M_1 and M_2 are of the same class, they exhibit similar number of fails and run time, and so do their induced models. We note that the search efforts for the models which use both M_1 and M_2 is greatly reduced. In the second and third groups, about 99% of the search effort is saved. This shows that there is much more constraint propagation than using single models alone. The fifth group exhibits further search reduction and always achieves the smallest number of fails, which is only half of those in the third group. Although these models are not the fastest, they are not far behind the



(a) $(18, 1.0)$



(b) $(18, 0.8)$



(c) $(18, 0.6)$

Figure 8: Average Run Times and Ratios of Run Times to the Single Models for the Series $(18, 1.0)$, $(18, 0.8)$, and $(18, 0.6)$ Respectively

best.

It must be remembered that in real problems (such as Langford's problem), the problem specification itself restricts the constraint formulations in a viewpoint. Here, we arbitrarily declare two different instances to be mutually redundant models of the same "problem" by forcing them to have equivalent sets of solutions with respect to the channeling constraints. This does not parallel how problem modeling is done in practice. Hence, we should not expect the large reduction in search from combining M_1, M_2, and/or their induced models that

Table 3: Comparison Results Using Pairs of Mutually Redundant Random Permutation CSPs

Models	Search Variables	$(18, 1.0, 0.21)$		$(18, 1.0, 0.22)$		$(18, 1.0, 0.23)$		$(18, 0.8, 0.26)$		$(18, 0.8, 0.27)$		$(18, 0.6, 0.32)$		$(18, 0.6, 0.33)$		$(18, 0.6, 0.34)$	
		Fails	Time	Fails	Time	Fails	Time	Fails	Time	Fails	Time	Fails	Time	Fails	Time	Fails	Time
M_1	X	36830	16.62	22251	10.58	12298	6.18	20485	9.98	13580	6.9	12800	6.61	9544	5.2	4682	2.7
$i(f, M_1)$	Y	37540	16.92	22152	10.5	13391	6.72	42403	18.77	27831	12.91	86632	35.16	63700	26.75	73150	29.51
M_2	Y	37621	16.89	22556	10.66	12834	6.42	20688	10.06	13565	6.82	13141	6.77	9652	5.25	4810	2.77
$i(f^{-1}, M_2)$	X	38133	17.13	22651	10.78	13444	6.79	42740	18.83	28065	12.8	88116	35.73	64229	26.8	75039	30.33
$M_1 \overset{C_c}{\bowtie} M_2$	X	351	0.42	255	0.34	176	0.25	263	0.35	203	0.28	203	0.31	170	0.28	134	0.23
$M_1 \overset{C_c}{\bowtie} M_2$	Y	354	0.42	256	0.34	185	0.26	261	0.35	200	0.29	198	0.31	171	0.28	135	0.23
$M_1 \overset{C_c}{\bowtie} M_2$	$X \cup Y$	304	0.39	228	0.32	159	0.24	229	0.33	174	0.27	180	0.29	153	0.26	105	0.19
$M_1 \sqcap i(f^{-1}, M_2)$	X	204	**0.18**	144	**0.14**	123	**0.1**	172	**0.16**	131	**0.12**	159	0.16	133	**0.14**	112	**0.12**
$i(f, M_1) \sqcap M_2$	Y	205	**0.18**	144	**0.14**	120	0.11	169	**0.16**	135	0.13	158	**0.15**	134	**0.14**	110	**0.12**
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	X	16834	16.66	10437	10.86	5906	6.65	11238	11.93	7575	8.55	8240	9.2	6202	7.35	3291	3.96
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	Y	17711	17.58	10497	10.96	6448	7.17	13610	14.02	9349	10.15	14624	14.9	10176	10.79	8015	8.83
$M_1 \overset{C_c}{\bowtie} i(f, M_1)$	$X \cup Y$	13840	14.3	8639	9.41	4916	5.66	10100	11.16	6927	7.95	7893	8.97	5958	7.11	3228	3.95
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	X	17970	17.82	10704	11.24	6454	7.18	13736	14.09	9422	10.2	14938	15.21	10331	10.97	8301	9.07
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	Y	17182	17.04	10549	11.06	6066	6.84	11335	12.11	7598	8.5	8422	9.39	6259	7.38	3343	4.03
$M_2 \overset{C_c}{\bowtie} i(f^{-1}, M_2)$	$X \cup Y$	14890	15.52	9024	9.83	5542	6.42	9650	10.56	6928	7.91	8320	9.4	5719	6.76	4053	4.98
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	X	127	0.25	114	0.22	100	0.18	116	0.22	100	0.2	99	0.2	97	0.19	68	0.16
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	Y	127	0.25	115	0.22	100	0.18	115	0.23	103	0.2	100	0.21	96	0.19	71	0.17
$M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$	$X \cup Y$	**119**	0.25	**106**	0.2	**88**	0.18	**108**	0.22	**92**	0.18	**94**	0.2	**88**	0.19	**57**	0.15

we find in random permutation problems to occur in real problems, in general. Nevertheless, our experiments are useful in demonstrating the extreme case of the benefit of combining mutually redundant models.

7 Concluding Remarks

We conclude the paper by giving discussions and summarizing our contributions and possible directions for future research.

7.1 Discussion

Handcrafting CSP models is an unamiable and costly task performed daily by human modelers, who should thus find model induction a useful tool. What human modelers need to be familiar with is the viewpoints of a problem and the channeling constraints between them. Moreover, the execution time for model induction is acceptable when constraint arities are small: the induced models used in all of our experiments can be generated almost instantly from the input models. The generated models can be combined with other models using model intersection and/or channeling. Model intersection helps increase constraint propagation within the same viewpoint by adding new constraints or making the existing constraints tighter, while model channeling provides an additional source of constraint propagation with the introduction of an extra viewpoint. Model redundancy is a relatively new concept and our formal study helps advance our understanding of its benefits. It demonstrates that mutually redundant models can incorporate different information and that combining the information from different sources can improve search efficiency. Our work is also a means to open up new possibilities to study, understand, and apply model redundancy in constraint satisfaction.

7.2 Contributions

The contributions of our work can be summarized as follows. First, based on the viewpoint notion, we have introduced model induction, which can automatically generate an alternative model in a different viewpoint from an existing model. We have given its syntactic construction rule, detailed examples, and also examined its properties. Second, we have proposed two methods to combine models, namely model intersection and model channeling. They allow combining two models of the same problem in the same and different viewpoints respectively. The latter is a formalization of redundant modeling by using the constraints of both models and defining a relationship between the viewpoints of the constituent models with the use of channeling constraints. Third, we have identified three new classes of combined models by exploiting different applications of model induction, intersection, and channeling. We have demonstrated the usefulness of the three classes of combined models using Langford's problem and random Permutation CSPs. Model induction, intersection, and channeling increase the flexibility of human modelers by producing more models for a problem and providing more possibilities of combining models. Benchmark results on permutation problems confirm that all the three proposed classes of combined models can achieve more propagation and better solving efficiency than the individual models. In fact, the model $M_{12^i} \overset{C_c}{\bowtie} M_{1^i2}$ in the third class, when available, always produces the fewest number of fails.

7.3 Future Work

Our work proposes a systematic study of transforming and combining CSP models. There is plenty of scope for future work on this important direction of research. First, it would be interesting to refine the definition of model induction. Although our current definition is applicable to general CSPs, our empirical results are developed for only Permutation CSPs. It will be interesting to check if the same techniques can be applied/generalized to other CSP classes. For example, in many assignment problems, it is possible to have two viewpoints, which use variables with integer domains and set domains respectively. Generalizing model induction to generate a set model from an integer model is a challenge. Further study can also be conducted to refine the requirement on the type of channeling constraints that model induction can apply.

Second, it would be preferable to avoid using the extensional representation of constraints, if possible. For many constraints, it is simply impracticable to construct the extensional representation, and it would be in any case preferable not to solve CSPs in extensional form since propagation is generally more time-consuming in many constraint programming systems if constraints are expressed extensionally rather than intensionally. It would be worthwhile to study how the indexical representation [12] of a constraint can be learned from its extensional counterpart. Dao *et al.* [12] proposed learning indexicals, which are approximations of constraints and enforce only bounds consistency. Cheng *et al.* [8, 7] proposed *box constraint collection* (BCC), which allows efficient representation of extensional constraints. BCC enforces arc consistency on the constraints and can be implemented efficiently using indexicals. Furthermore, SICStus Prolog [2] also provides the `case/4` constraint for representing arbitary constraints. These tools can be used together with model induction to represent and propagate the extensional constraints in the induced models more efficiently.

Acknowledgments

The work described in this paper was substantially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK4131/05E, CUHK4219/04E, and CUHK4358/02E). The third author is supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075. We would like to thank Ho-fung Leung, Evangeline Young, Peter Stuckey, and the anonymous referees from AAAI'02 and the *Constraints* journal for their comments which helped improve the paper undoubtedly.

References

[1] *ILOG Solver 4.4 Reference Manual*, 1999.

[2] *SICStus Prolog User's Manual*, 2003.

[3] C. Ansótegui and F. Manyà. Mapping problems with finite-domain variables into problems with Boolean variables. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing*, pages 1–15, 2004.

[4] C. Bessière, J.-C. Régin, R.H.C. Yap, and Y. Zhang. An optimal coarse-grained arc consistency algorithm. *Artificial Intelligence*, 165(2):165–185, 2005.

[5] 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. *Constraints*, 4(2):167–192, 1999.

[6] B.M.W. Cheng, J.H.M. Lee, and J.C.K. Wu. Speeding up constraint propagation by redundant modeling. In *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming*, pages 91–103, 1996.

[7] K.C.K. Cheng, J.H.M. Lee, and P.J. Stuckey. Box constraint collections for adhoc constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, pages 214–228, 2003.

[8] K.C.K. Cheng, J.H.M. Lee, and P.J. Stuckey. Efficient representation of ad-hoc constraints. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1368–1369, 2003.

[9] C.W. Choi, J.H.M. Lee, and P.J. Stuckey. Propagation redundancy for permutation channels. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1370–1371, 2003.

[10] C.W. Choi, J.H.M. Lee, and P.J. Stuckey. Propagation redundancy in redundant modelling. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, pages 229–243, 2003.

[11] C.W. Choi, J.H.M. Lee, and P.J. Stuckey. Removing propagation redundant constraints in redundant modeling. *ACM Transactions on Computational Logic*, 8(4):(to appear), 2007.

[12] T.B.H. Dao, A. Lallouet, A. Legtchenko, and L. Martin. Indexical-based solver learning. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 541–556, 2002.

[13] I. Dotú, A. del Val, and M. Cebrián. Redundant modeling for the quasigroup completion problem. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, pages 288–302, 2003.

[14] P. Flener, A.M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 462–476, 2002.

[15] P. Flener, B. Hnich, and Z. Kiziltan. Compiling high-level type constructors in constraint programming. In *Proceedings of the 3rd International Symposium on Practical Aspects of Declarative Languages*, pages 229–244, 2001.

[16] A. Frisch, T. Peugniez, A. Goggett, and P. Nightingale. Solving non-Boolean satisfiability problems with stochastic local search: A comparison of encodings. *Journal of Automated Reasoning*, 35(1–3):143–179, 2005.

[17] A.M. Frisch, M. Grum, C. Jefferson, B.M. Hernández, and I. Miguel. The design of ESSENCE: A constraint language for specifying combinatorial problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 80–87, 2007.

[18] A.M. Frisch, C. Jefferson, B.M. Hernández, and I. Miguel. The rules of constraint modelling. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 109–116, 2005.

[19] P.A. Geelen. Dual viewpoint heuristics for binary constraint satisfaction problems. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 31–35, 1992.

[20] I.P. Gent. Arc consistency in SAT. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 121–125, 2002.

[21] B. Hnich. *Function Variables for Constraint Programming*. PhD thesis, Department of Information Science, Uppsala University, 2003.

[22] B. Hnich, B.M. Smith, and T. Walsh. Dual modelling of permutation and injection problems. *Journal of Artificial Intelligence Research*, 21:357–391, 2004.

[23] J. Jourdan. *Concurrent constraint multiple models in CLP and CC languages: Toward a programming methodology by modelling*. PhD thesis, Université Denis Diderot, Paris VII, 1995.

[24] J. Jourdan. Concurrent constraint multiple models in CLP and CC languages: Toward a programming methodology by modelling. In *Proceedings of the INFORMS Conference*, 1995.

[25] Y.C. Law and J.H.M. Lee. Model induction: a new source of CSP model redundancy. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 54–60, 2002.

[26] Y.C. Law and J.H.M. Lee. Breaking value symmetries in matrix models using channeling constraints. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing*, pages 375–380, 2005.

[27] Y.C. Law and J.H.M. Lee. Symmetry breaking constraints for value symmetries in constraint satisfaction. *Constraints*, 11(2–3):221–267, 2006.

[28] Y.C. Law, J.H.M. Lee, and B.M. Smith. Generating a pair of mutually redundant random permutation CSPs. Technical report, Computer Science & Engineering, CUHK and Cork Constraint Computation Centre, UCC, 2005.

[29] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[30] K. Marriott and P.J. Stuckey. *Programming with Constraints*, chapter 8, pages 266–271. The MIT Press, 1998.

[31] B. Martínez-Hernández and A.M. Frisch. The automatic generation of redundant representations and channelling constraints. In *Proceedings of the 5th International Workshop on Constraint Modelling and Reformulation*, pages 42–56, 2006.

[32] F. Rossi, C. Petrie, and V. Dhar. On the equivalence of constraint satisfaction problems. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 550–556, 1990.

[33] O. Roussel. Some notes on the implementation of *csp2sat+zchaff*, a simple translator from CSP to SAT. In *Proceedings of the 2nd International Workshop on Constraint Propagation and Implementation*, pages 83–88, 2005.

[34] B.M. Smith. Modelling a permutation problem. Technical Report 2000.18, School of Computer Studies, University of Leeds, 2000.

[35] B.M. Smith. Dual models in permutation problems. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 615–619, 2001.

[36] B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81(1-2):155–181, 1996.

[37] T. Walsh. SAT v CSP. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming*, pages 441–455, 2000.

[38] T. Walsh. Permutation problems and channelling constraints. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, pages 377–391, 2001.

[39] R. Weigel and C. Bliek. On reformulation of constraint satisfaction problems. In *Proceedings of 13th European Conference on Artificial Intelligence*, pages 254–258, 1998.