Automatic Dominance Breaking for a Class of Constraint Optimization Problems

Jimmy H.M. Lee and **Allen Z. Zhong**
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{jlee, zwzhong}@cse.cuhk.edu.hk

Abstract

Exploiting dominance relations in many Constraint Optimization Problems can drastically speed up the solving process in practice. Identification and utilization of dominance relations, however, usually require human expertise. We present a theoretical framework for a useful class of constraint optimization problems to detect dominance automatically and formulate the generation of the associated dominance breaking nogoods as constraint satisfaction. By controlling the length and quantity of the nogoods, our method can generate dominance breaking nogoods of varying strengths. Experimentation confirms runtime improvements of up to three orders of magnitude against manual methods.

1 Introduction

Dominance relations in Constraint Optimization Problems (COPs) describe relations between two assignments where one is known to be subordinate compared with another with respect to satisfiability and/or objective value. Once dominance relations are found in a COP, they can be used to speed up Branch and Bound (BnB) search [Ibaraki, 1977].

A wealth of research studies how to exploit dominance relations in practice. With sophisticated insights of the problem structure, one could either add *dominance breaking constraints* to exclude dominated assignments [Prestwich and Beck, 2004; Getoor *et al.*, 1997] or augment the BnB search algorithm with *dominance test* [Monette *et al.*, 2007; Aldowaisan, 2001; Korf, 2004]. Empirical evidence has shown that these problem-specific methods can dramatically reduce the search space and so speed up the solving process.

Dominance breaking is useful in solving COPs, but identification of dominance relations is non-trivial in general. Chu and Stuckey [2012] give the first generic method for identifying a class of dominance relations and generating dominance breaking constraints for COPs. While clever and interesting, the method calls for manual derivation of mappings and conditions, which is taxing for the mathematically uninclined. Mears and de la Banda [2015] automate the derivation process to a large extent, but their method still has to select symmetries manually. Also, they could generate only the same constraints as those reported in the literature.

Manual derivation of free form dominance breaking constraints requires deep understanding of the problems and sometimes even ingenuity. Our goal is to automate the process to make dominance breaking accessible to the non-experts by restricting the form of the dominance breaking constraints to nogoods. We start by giving a general theory of dominance relations over constraints and specialize our attention to partial assignments, which can be interpreted as conjunctions of equality constraints. Theorems are given to identify the sufficient conditions of when an assignment constraint dominates another for COPs with objectives and constraints of certain useful forms. Negation of dominated partial assignments are dominance breaking nogoods. Most importantly, nogoods generation can be formulated mechanically as constraint satisfaction. By controlling the length and quantities of the nogoods, our method can tune the collective strengths of the generated nogoods and find pruning opportunities that are often neglected by manual methods. Experimentation on a diversified set of benchmarks confirms the practicality of our method, which compares favourably against manual methods both in efficiency and ease of use, and exhibits runtime (including both nogood generation and solving time) speed-up of up to three orders of magnitude.

2 Background

A *Constraint Satisfaction Problem (CSP)* P is a tuple (X, D, C) consisting of a finite set of variables $X = \{x_1, \ldots, x_n\}$, a mapping D from variable $x \in X$ to its finite domain $D(x)$ and a set of constraints C, where each constraint $c \in C$ is a subset of the Cartesian product $D(x_{i_1}) \times \cdots \times D(x_{i_k})$ over the *scope* $var(c) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$. A *Constraint Optimiaztion Problem (COP)* (X, D, C, f) extends a CSP with an objective function f.

An *assignment* $x = v$ is an equational constraint which assigns a value $v \in D(x)$ to variable $x \in X$. A *partial assignment θ for variables $X' \subseteq X$* is the set of assignments $\{x = v | x \in X'\}$, where X' is the *scope* $var(\theta)$ of θ. A *full assignment* is a partial assignment when $X' = X$. When the context is clear, we use θ to denote the tuple $(v_{i_1}, \ldots, v_{i_l})$, where $i_j < i_{j+1}$, $x_{i_j} \in X'$ and $|X'| = l$. The *projection $\theta{\downarrow_Y}$ of θ onto $Y \subseteq var(\theta)$* is the partial assignment $\{x = v | (x = v) \in \theta \wedge x \in Y\}$.

A full/partial assignment θ *satisfies* a constraint c iff $\theta{\downarrow_{var(c)}} \in c$, where $var(c) \subseteq var(\theta)$. A *solution* of a COP

$P = (X, D, C, f)$ is a full assignment that satisfies all constraints in C, and P is *satisfiable* if the the set of all solutions $sol(P)$ is not empty. In the remainder of the paper, we use $\bar{\theta}$ to emphasize that it is a full assignment.

The objective function f maps every full assignment to a real number. Without loss of generality, the goal of solving a COP is to find an *optimal solution* $\bar{\theta}_{opt}$ such that $\bar{\theta}_{opt}$ is a solution of P and $f(\bar{\theta}_{opt}) \leq f(\bar{\theta}')$ for any other solution $\bar{\theta}'$ of P. In other words, the objective function is minimized. (Remove) In constraint programming, The Branch-and-bound (BnB) algorithm [Land and Doig, 1960] usually solves a COP by iteratively finding a solution with better objective value, and pruning subtrees that contain no better solution than the best solution found so far. The iterative process is repeated until no better solution can be found and the last solution is returned as the optimal solution.

Here we review some basic concepts in relational mathematics. A *binary relation* R over two sets S and S' is a set of ordered pairs (s, s') where $s \in S$ and $s' \in S'$. If $S = S'$, such a binary relation is *homogeneous*; otherwise it is *heterogeneous*. For a homogeneous relation R over S, R is *transitive* if $\forall s, s', s'' \in S, sRs' \wedge s'Rs'' \Rightarrow sRs''$, and it is *irreflexive* if $\nexists s \in S$ such that sRs. The *transitive closure* of a homogeneous relation R on a set S is the smallest relation on S that contains R and is transitive. In our work, we slightly generalise the transitive closure to heterogeneous relation $R : S \mapsto S'$ by a two-step construction: (1) construct a relation $\hat{R}$ over the union $\hat{S} = S \cup S'$ such that $\hat{R} = R$ as a set of ordered pairs, and 2) take the transitive closure $\hat{R}^+$ of $\hat{R}$ on the set $\hat{S}$.

Let Θ^P be the set of full assignments of a COP $P = (X, D, C, f)$. A *dominance relation* $\prec_a$ *with respect to* P is a transitive and irreflexive relation such that for $\bar{\theta}, \bar{\theta}' \in \Theta^P$ if $\bar{\theta} \prec_a \bar{\theta}'$ with respect to P, then either: 1) $\bar{\theta}$ is a solution of P and $\bar{\theta}'$ is not a solution of P, or 2) both $\bar{\theta}$ and $\bar{\theta}'$ are solutions of P and $f(\bar{\theta}) \leq f(\bar{\theta}')$, or 3) both $\bar{\theta}$ and $\bar{\theta}'$ are not solutions of P and $f(\bar{\theta}) \leq f(\bar{\theta}')$. We say that $\bar{\theta}$ *dominates* $\bar{\theta}'$. The following theorem states that it is sound to prune all dominated full assignments.

Theorem 1. *[Chu and Stuckey, 2012] Given a COP $P = (X, D, C, f)$ and a dominance relation $\prec_a$ with respect to P. We can prune all full assignments $\bar{\theta}' \in \Theta^P$ whenever $\exists \bar{\theta} \in \Theta^P$ such that $\bar{\theta} \prec_a \bar{\theta}'$, without changing the satisfiability or optimal value of P.*

Note that a dominance relation should be both transitive and irreflexive. If P is satisfiable, then there is at least one optimal solution $\bar{\theta}_{opt}$ of P that is not dominated by any other solutions. Consider a simple COP P with one variable x with domain $\{0, 1\}$ and a constant objective function, where Both $\{x = 0\}$ and $\{x = 1\}$ are optimal solutions of P. Given a dominance relation $\prec_a$ with respect to P, either $\{x = 1\} \prec_a \{x = 0\}$ or $\{x = 0\} \prec_a \{x = 1\}$, but not both. Otherwise, we will have $\{x = 1\} \prec_a \{x = 1\}$ due to the transitive property, but it violates the irreflexive property of $\prec_a$.

3 Automatic Generation of Dominance Breaking Nogoods

This section gives an overview of our method. Different from other works in the literature where dominance relations are manually identified, our method can automate the process of identifying and exploiting dominance relations as follows:

1. Given the problem model of a COP P, we will first analyze the objective and constraints of P and build a generation model as a CSP mechanistically for identifying all dominance breaking constraints with respect to P.
2. When instance data is presented, all solutions of the generation model are searched, and each solution corresponds to a dominance breaking constraint, which is in the form of a nogood.
3. All generated nogoods are added to the problem model of P before solving.

The key step is to build the generation model automatically based on the problem model of P. In the rest of the paper, our focus is to show that, for a COP P, the constraints in the generation model can be obtained directly from constraints and objectives in P. The generation model can be obtained by using only one pass automatic analysis on the problem model of P.

To facilitate the presentation of our work, we will first present a general theory of dominance relations over constraints in Section 4. If a constraint c is dominated by some constraint c' with respect to a COP P, the negation $\neg c$ is a dominance breaking constraint for P. Further, we will show that it suffices to consider three conditions, namely *irreflexivity*, *betterment* and *implied satisfaction*, to establish the dominance relation between c and c'. In Section 5, we will restrict our attention to dominance relations over *assignment constraints*. We will give concrete results on what constraints should be presented in the generation model for different objectives and constraints in the problem model. In Section 6, we will give the skeleton code for implementation.

4 Dominance Relations Between Constraints

Chu and Stuckey [2012] investigated dominance relations on full assignments and search nodes. It is straightforward to generalise it to dominance relations between constraints.

Given a COP $P = (X, D, C, f)$. Let Θ_c denote the set of all full assignments satisfying c. Suppose there are two constraints c and c' of the same scope $X' \subseteq X$, we say c *dominates* c', i.e. $c \prec c'$, with respect to P if $\forall \bar{\theta}' \in \Theta_{c'}, \exists \bar{\theta} \in \Theta_c$, such that $\bar{\theta} \prec_a \bar{\theta}'$ for some dominance relation $\prec_a$ with respect to P. Note that we do not require $c, c' \in C$. The following theorem is a direct consequence of Theorem 1.

Theorem 2. *Given a COP $P = (X, D, C, f)$ and two constraints c and c' over $X' \subseteq X$. If $c \prec c'$ with respect to P, then P has the same satisfiability or optimal value as $P' = (X, D, C \cup \{\neg c'\}, f)$*

Proof. $\forall \bar{\theta}' \in \Theta_{c'}, \exists \bar{\theta} \in \Theta_c$ s.t. $\bar{\theta} \prec_a \bar{\theta}'$. By Theorem 1, we can prune all dominated full assignments in $\Theta_{c'}$ by adding $\neg c'$ to the constraint set C of P. □

Once we could establish $c \prec c'$ with respect to P, the negation of c' is the desired *dominance breaking constraint*. To check $c \prec c'$, we can simply compare all solutions in Θ_c against those in $\Theta_{c'}$, but this is impractical. We give a sufficient condition for $c \prec c'$.

Theorem 3. *Given a COP $P = (X, D, C, f)$ and two constraints c and c' over $X' \subseteq X$. If there exists a bijective mapping $\sigma : \Theta_c \mapsto \Theta_{c'}$ such that:*

- *irreflexivity: the transitive closure of σ is irreflexive*
- *betterment: $\forall \bar{\theta} \in \Theta_c$, $f(\bar{\theta}) \leq f(\sigma(\bar{\theta}))$*
- *implied satisfaction: $\forall \bar{\theta} \in \Theta_c$, $\sigma(\bar{\theta})$ is a solution of P implies $\bar{\theta}$ is a solution of P*

then $c \prec c'$.

Proof. (Sketch) We need to show that $\forall \bar{\theta}' \in \Theta_{c'}, \exists \bar{\theta} \in \Theta_c$ such that $\bar{\theta} \prec_a \bar{\theta}'$ where $\prec_a$ is a dominance relation with respect to P. The idea is similar to the proof of Theorem 3 by Chu and Stuckey [2012], based on which we show the transitive closure of σ is indeed a dominance relation $\prec_a$.

We further prove that $\forall \bar{\theta}' \in \Theta_{c'}, \exists \bar{\theta} \in \Theta_c$ such that $\bar{\theta} \prec_a \bar{\theta}'$ with respect to P. Note that σ is a bijective mapping, so $\forall \bar{\theta}' \in \Theta_{c'}, \exists \sigma^{-1}(\bar{\theta}') \in \Theta_c$. Since $\prec_a$ is constructed by taking the transitive closure of σ, $\sigma^{-1}(\bar{\theta}') \prec_a \bar{\theta}'$. Hence, $c \prec c'$. □

From now on, when the context is clear, we use $\prec$ to denote a dominance relation between full assignments or constraints. To generate dominance breaking constraints, it remains to construct an appropriate σ for the purpose. In the following section, we show that this is possible for certain classes of objectives and constraints in the COP by limiting our attention to dominance relation between *assignment constraints*.

5 Dominance and Assignment Constraints

The bijective mapping in Theorem 3 needs to satisfy the irreflexivity, betterment and implied satisfaction conditions. Fulfilling irreflexivity turns out to be simple.

Theorem 4. *Given a COP $P = (X, D, C, f)$, and two constraints c and c' over $X' \subseteq X$. If $\Theta_c \cap \Theta_{c'} = \emptyset$, all mappings $\sigma : \Theta_c \mapsto \Theta_{c'}$ and their transitive closures are irreflexive.*

Proof. Since $\Theta_c \cap \Theta_{c'} = \emptyset$, $\theta \neq \theta' \forall (\theta, \theta') \in \sigma$. If $(\bar{\theta}, \bar{\theta}') \in \sigma$, then $(\bar{\theta}', *) \notin \sigma$ where $*$ is any full assignment. Thus, the transitive closure of σ is also irreflexive. □

Fulfilling betterment and implied satisfaction for general constraints is difficult, but we consider a special class of constraints which arises out of partial assignments. A partial assignment θ with scope F is technically not a constraint, but essentially a set of equations, the conjunction of which, i.e. $\bigwedge_{e \in \theta}(e)$, can be interpreted as a constraint. We call it an *assignment constraint*. To avoid confusion in notation, we use $AC(\theta)$ to denote an assignment constraint for the partial assignment θ. The scope of $AC(\theta)$ is also F, and the *length* is $|F|$. We are interested in the conditions when an assignment constraint $AC(\theta)$ dominates another $AC(\theta')$ with the same scope. Note that $\neg AC(\theta')$ is in the form of a nogood [Katsirelos and Bacchus, 2005]. We call it a *dominance breaking nogood* with respect to P.

If $\theta \neq \theta'$, then $\Theta_{AC(\theta)} \cap \Theta_{AC(\theta')} = \emptyset$ and irreflexivity in Theorem 3 holds. What remains is to find a bijective mapping σ that satisfies betterment and implied satisfaction.

Suppose there are two assignment constraints $AC(\theta)$ and $AC(\theta')$ with the same scope $F \subseteq X$. We define the *mutation mapping* as $\sigma_m : \Theta_{AC(\theta)} \mapsto \Theta_{AC(\theta')}$, which maps a full assignment $\bar{\theta} \in \Theta_{AC(\theta)}$ to $\sigma_m(\bar{\theta}) \in \Theta_{AC(\theta')}$ such that $\sigma_m(\bar{\theta}) = (\bar{\theta} \backslash \theta) \cup \theta'$. We note that every full assignment $\bar{\theta} \in \Theta_{AC(\theta)}$ has the form $\bar{\theta} = \theta \cup \{x = v | x \notin F \wedge v \in D(x)\}$. Thus, $\sigma_m(\bar{\theta}) = \theta' \cup \{x = v | x \notin F \wedge v \in D(x)\}$, which "mutates" the θ component of $\bar{\theta}$ to become θ'. The following proposition shall be useful in our later discussion.

Proposition 1. *Given a mutation mapping $\sigma_m : \Theta_{AC(\theta)} \mapsto \Theta_{AC(\theta')}$. $\forall \bar{\theta} \in \Theta_{AC(\theta)}$, $\bar{\theta} \backslash \theta = \sigma_m(\bar{\theta}) \backslash \theta'$.*

In the following, we give sufficient conditions for different classes of objectives and constraints to fulfill betterment and implied satisfaction. Unless otherwise stated, **all formal results in the remainder of this section are given with the following context**: *given a COP $P = (X, D, C, f)$ and two partial assignments θ and θ' with the same scope F and the associated mutation mapping $\sigma_m : \Theta_{AC(\theta)} \mapsto \Theta_{AC(\theta')}$.*

5.1 The Betterment Condition

Now we show the sufficient conditions for betterment hold for σ_m, namely $\forall \bar{\theta} \in \Theta_{AC(\theta)}, f(\bar{\theta}) \leq f(\sigma_m(\bar{\theta}))$. The key idea is to define the *projection $f{\downarrow_{var(\theta)}}$ of f onto $var(\theta)$* so that the relative magnitude of $f(\bar{\theta})$ and $f(\sigma_m(\bar{\theta}))$ can be determined. In the following, we consider two types of objectives: separable and supermodular/submodular functions.

Separable Objectives

A function f is *separable* if it can be written as a linear combination of functions of individual variables, i.e. for a full assignment $\bar{\theta} = (v_1, \ldots, v_n)$, $f(\theta) = f_1(v_1) + \cdots + f_n(v_n)$, where each component is $f_i : \mathbb{Z} \mapsto \mathbb{R}$. For a partial assignment θ, we define $f{\downarrow_{var(\theta)}}(\theta) = f_{i_1}(v_{i_1}) + \cdots + f_{i_l}(v_{i_l})$ where $(x_{i_j} = v_{i_j}) \in \theta$ for $j = 1, \ldots, l$.

Theorem 5. *Suppose the objective f is a separable function. If $f{\downarrow_F}(\theta) \leq f{\downarrow_F}(\theta')$, then $\forall \bar{\theta} \in \Theta_{AC(\theta)}$, $f(\bar{\theta}) \leq f(\sigma_m(\bar{\theta}))$.*

Proof. Suppose $\bar{\theta} = (v_1, \ldots, v_n) \in \Theta_{AC(\theta)}$ and $\sigma_m(\bar{\theta}) = (v'_1, \ldots, v'_n)$, then we have

$$f(\bar{\theta}) = f{\downarrow_F}(\theta) + \sum_{x_{i_j} \notin F} f_{i_j}(v_{i_j})$$

$$f(\sigma_m(\bar{\theta})) = f{\downarrow_F}(\theta') + \sum_{x_{i_j} \notin F} f_{i_j}(v'_{i_j})$$

by grouping the terms. By Proposition 1, $\bar{\theta} \backslash \theta = \sigma_m(\bar{\theta}) \backslash \theta'$, and $\sum_{x_{i_j} \notin F} f_{i_j}(v_{i_j}) = \sum_{x_{i_j} \notin F} f_{i_j}(v'_{i_j})$. Thus $f(\bar{\theta}) \leq f(\sigma_m(\bar{\theta})) \iff f{\downarrow_F}(\theta) \leq f{\downarrow_F}(\theta')$. □

A typical example of separable functions is the linear function $f(\bar{\theta}) = \sum_i w_i v_i$, where $w_i \in \mathbb{R}$.

Supermodular and Submodular Objectives

A *supermodular function* is a set function $g : 2^V \mapsto \mathbb{R}$ that assigns each subset $S \subseteq V$ a value $g(S) \in \mathbb{R}$ such that

$$g(S \cup T) - g(S) \leq g(S' \cup T) - g(S')$$

for every $S, S' \subseteq V$ with $S \subseteq S'$ and $T \subseteq V \backslash S'$. We could treat a supermodular function as an objective in a binary COP $P = (X, D, C, f)$ where $\forall x \in X, D(x) = \{0, 1\}$. A full assignment $\bar{\theta}$ can be associated with a set $S(\bar{\theta}) = \{i | (x_i = 1) \in \bar{\theta}\}$, and we say *$f$ is equivalent to a supermodular function g* if $f(\bar{\theta}) = g(S(\bar{\theta}))$. Similarly for a partial assignment θ, $S(\theta) = \{i | (x_i = 1) \in \theta\}$, and we define $f{\downarrow_{var(\theta)}}(\theta) = g(S(\theta))$

Theorem 6. *Suppose the objective f is equivalent to a supermodular function g. If $f{\downarrow_F}(\theta) \leq f{\downarrow_F}(\theta')$ and $S(\theta) \subseteq S(\theta')$, then $\forall \bar{\theta} \in \Theta_{AC(\theta)}$, $f(\bar{\theta}) \leq f(\sigma_m(\bar{\theta}))$.*

Proof. Since g is a supermodular,

$$g(S(\theta) \cup T) - g(S(\theta)) \leq g(S(\theta') \cup T) - g(S(\theta'))$$

for $S(\theta) \subseteq S(\theta') \subseteq V = \{1, \ldots, n\}$ and any set $T \subseteq V \backslash S(\theta')$. For a full assignment $\bar{\theta} \in \Theta_{AC(\theta)}$, we let $T = \{i | (x_i = 1) \in \bar{\theta} \wedge x_i \notin F\}$. Since $\bar{\theta} = \theta \cup \{x = v | x \notin F \wedge v \in D(x)\}$, we have $f(\bar{\theta}) = g(S(\theta) \cup T)$. Furthermore, since $\bar{\theta}' = \sigma_m(\bar{\theta}) = (\bar{\theta} \backslash \theta) \cup \theta'$, $f(\bar{\theta}') = g(S(\theta') \cup T)$. Then

$$f(\bar{\theta}) \leq f(\bar{\theta}') - g(S(\theta')) + g(S(\theta))$$
$$\iff f(\bar{\theta}) \leq f(\bar{\theta}') - f{\downarrow_F}\ (\theta') + f{\downarrow_F}\ (\theta)$$

Since $f{\downarrow_F}\ (\theta) \leq f{\downarrow_F}\ (\theta')$, the above inequality implies that $f(\bar{\theta}) \leq f(\bar{\theta}')$. □

A function g is *submodular* if $-g$ is supermodular. Thus minimizing a supermodular function is equivalent to maximizing a submodular function. We note that Theorems 3 and 6 (and also Theorem 5) can be easily adapted for maximization problems. Linear functions are also examples of submodular functions. Another typical example is the cut function in a graph with non-negative weights on edges.

5.2 The Implied Satisfaction Condition

Now we consider the implied satisfaction condition in Theorem 3. We give sufficient conditions for when $\forall \bar{\theta} \in \Theta_{AC(\theta)}, \sigma_m(\bar{\theta}) \in sol(P)$ implies $\bar{\theta} \in sol(P)$. Note that a full assignment $\bar{\theta} \in sol(P)$ if $\bar{\theta}$ satisfies all $c \in C$. It suffices to consider each constraint c separately.

We say a partial assignment θ is *applied to* $c \in C$ by replacing every occurrence of x in c by value v for all $(x = v) \in \theta$ where $x \in var(c) \cap var(\theta)$. The resulting constraint $c\theta$ has scope $var(c) \backslash var(\theta)$. The following proposition is useful to prove implied satisfaction for various constraints in later subsections.

Proposition 2. *$\forall \bar{\theta} \in \Theta_{AC(\theta)}$ and a constraint $c \in C$, if $c\theta' \Rightarrow c\theta$, then $\sigma_m(\bar{\theta})$ satisfies $c \Rightarrow \bar{\theta}$ satisfies c.*

Proof. By Proposition 1, let $\beta = \bar{\theta} \backslash \theta = \sigma_m(\bar{\theta}) \backslash \theta'$. Since $c\theta' \Rightarrow c\theta$, $c\theta' \subseteq c\theta$. Thus, β satisfies $c\theta' \Rightarrow \beta$ satisfies $c\theta$, which means $\sigma_m(\bar{\theta})$ satisfies $c \Rightarrow \bar{\theta}$ satisfies c. □

The types of constraints we will study include domain constraints, linear inequalities, Boolean disjunctions and the alldifferent and alldifferent_except_0 constraints.

Domain Constraints

A domain constraint $c = (x \in D(x))$ is a unary constraint which restricts a variable x to take values from a set $D(x)$. The condition for implied satisfaction is straightforward.

Theorem 7. *Given a domain constraint $c = (x \in D(x))$ and a full assignment $\bar{\theta} \in \Theta_{AC(\theta)}$. If either (a) $x \notin F$ or (b) $(x \in F)$ and $v \in D(x)$ for some $(x = v) \in \theta$, then $c\theta$ implies $c\theta'$.*

Proof. Suppose $x \notin F$. Then $c\theta = c = c\theta'$, which means $\sigma_m(\bar{\theta})$ satisfies c iff $\bar{\theta}$ satisfies c. Suppose $x \in F$ and $v \in D(x)$ for some $(x = v) \in \theta$. In this case, $c\theta$ is always true. Thus $\forall \bar{\theta} \in \Theta_{AC(\theta)}$, $\bar{\theta}$ always satisfies c. □

Linear Inequality Constraints

A linear inequality constraint has the form $c = (\sum w_i x_i \leq b)$ where $w_i, b \in \mathbb{R}$. The sufficient condition for implied satisfaction is stated as follows.

Theorem 8. *Given a linear inequality constraint $c = (\sum w_i x_i \leq b)$. If $e \leq e'$ where $e = \sum_{(x_i = v_i) \in \theta} w_i v_i$ and $e' = \sum_{(x_i = v_i') \in \theta'} w_i v_i'$, then $c\theta'$ implies $c\theta$.*

Proof. $c\theta = (\sum_{(x_i \notin F} w_i x_i \leq b - e)$ and $c\theta' = (\sum_{x_i \notin F} w_i x_i \leq b - e')$. Since $e \leq e'$, $b - e \geq b - e'$. Thus, $c\theta'$ implies $c\theta$. □

Boolean Disjunctions

The *Boolean disjunction* constraint $\vee_{x \in S} x$ requires at least one Boolean variable $x \in S$ takes the true value.

Theorem 9. *Given a Boolean disjunction constraint $c = (\vee x_i)$. If e' implies e where $e = \vee_{(x_i = v_i) \in \theta} v_i$ and $e' = \vee_{(x_i = v_i') \in \theta'} v_i'$, then $c\theta'$ implies $c\theta$.*

The proof idea is similar to that of Theorem 8.

Alldifferent and Alldifferent_except_0 Constraints

The *alldifferent* constraint [Régin, 1994] enforces that all variables in a set S take distinct values.

We say a partial assignment θ has *no duplicated values* if $\forall e, e' \in \theta$ where $e = (x_i = v_i), e' = (x_{i'} = v_{i'})$, $v_i \neq v_{i'}$. The sufficient condition for implied satisfaction requires the set of assigned values to variables in S to be the same.

Theorem 10. *Given a constraint $c = alldifferent(S)$ where $S \subseteq X$. Let $V = \{v | (x = v) \in \theta \wedge x \in F'\}$ and $V' = \{v' | (x = v') \in \theta' \wedge x \in F'\}$ where $F' = F \cap var(c)$. If $V = V'$ and both θ and θ' have no duplicated values, then $c\theta'$ implies $c\theta$.*

Proof. The alldifferent constraint can be expressed as $c = alldifferent(\{x | x \in var(c) \backslash F\} \cup \{x | x \in var(c) \cap F\})$. Since $V = V'$,

$$\begin{aligned} c\theta &= alldifferent(\{x | x \in var(c) \backslash F\} \cup V) \\ &= alldifferent(\{x | x \in var(c) \backslash F\} \cup V') \\ &= c\theta' \end{aligned}$$

Thus, $c\theta' \iff c\theta$. □

The alldifferent_except_0 constraint [Beldiceanu *et al.*, 2010] is a special case of the alldifferent constraint where all variables in a set S are required to take distinct values except for those variables that are assigned value 0. We say a partial assignment θ *has no duplicated values except 0* if $\forall(x_i = v_i), (x_{i'} = v_{i'}) \in \theta$ such that $v \neq v_{i'} \vee v = 0 \vee v_{i'} = 0$. The sufficient condition is also related to the set of assigned values of θ and θ' to variables in S.

Theorem 11. *Given $c = alldifferent\_except\_0(S)$ where $S \subseteq X$. Let $V = \{v|(x = v) \in \theta \wedge x \in F' \wedge v \neq 0\}$ and $V' = \{v'|(x = v') \in \theta' \wedge x \in F' \wedge v' \neq 0\}$ where $F' = F \cap var(c)$. If both θ and θ' have no duplicate values except 0 and $V \subseteq V'$, then $c\theta'$ implies $c\theta$.*

The proof idea is similar to that of Theorem 10.

5.3 Compatibility between Dominance Nogoods

So far we only consider the soundness of the generated dominance nogoods for a COP P. When all generated dominance breaking nogoods are added to P, we need to ensure that not all optimal solutions are eliminated. We now show a sufficient condition for preservation of the optimal value of P.

Given two partial assignments $\theta = (v_{i_1}, \ldots, v_{i_l})$ and $\theta' = (v'_{i_1}, \ldots, v'_{i_l})$ with the same scope F, we say θ is *lexicographically smaller* than θ', denoted as $\theta <_{lex} \theta'$, if there exists $x_{i_j} \in F$ such that $v_{i_j} < v'_{i_j}$ and $v_{i_{j'}} = v'_{i_{j'}} \forall j' < j$. We say θ *surpasses* θ' if either (a) $f{\downarrow_{var(\theta)}}(\theta) < f{\downarrow_{var(\theta')}}(\theta')$ or (b) $f{\downarrow_{var(\theta)}}(\theta) = f{\downarrow_{var(\theta')}}(\theta') \wedge \theta <_{lex} \theta'$. Note that the lexicographical ordering is a *total order*. The following theorem states that it is sufficient to enforce the condition that θ surpasses θ' so that adding all identified dominance breaking nogoods $\neg AC(\theta')$ to P will not change the satisfiability or optimal value.

Theorem 12. *Given f is either a separable or supermodular function. If $AC(\theta) \prec AC(\theta')$ and θ surpasses θ', the lexicographically smallest optimal solution satisfies $\neg AC(\theta')$.*

Proof. Suppose $f{\downarrow_{var(\theta)}}(\theta) < f{\downarrow_{var(\theta')}}(\theta')$. We claim that any solution satisfying $AC(\theta')$ is not optimal.

- If f is separable, then

$$f(\bar{\theta}) = f{\downarrow_{var(\theta)}}(\theta) + f{\downarrow_{X\setminus var(\theta)}}(\bar{\theta}\backslash\theta)$$

  Since $\bar{\theta}'\backslash\theta' = \bar{\theta}\backslash\theta$, any solution satisfying $AC(\theta')$ cannot be optimal since

$$\begin{aligned} f(\bar{\theta}') &= f{\downarrow_{var(\theta')}}(\theta') + f{\downarrow_{X\setminus var(\theta')}}(\bar{\theta}'\backslash\theta') \\ &= f{\downarrow_{var(\theta')}}(\theta') + f{\downarrow_{X\setminus var(\theta)}}(\bar{\theta}\backslash\theta) \\ &> f{\downarrow_{var(\theta)}}(\theta) + f{\downarrow_{X\setminus var(\theta)}}(\bar{\theta}\backslash\theta) \\ &> f(\bar{\theta}) \end{aligned}$$

- If f is supermodular, by theorem 6 we have $S(\theta) \subseteq S(\theta')$. So,

$$\begin{aligned} f(\bar{\theta}') &\geq f(\bar{\theta}) + g(S(\theta')) - g(S(\theta)) \\ &\geq f(\bar{\theta}) + f{\downarrow_{var(\theta')}}(\theta') - f{\downarrow_{var(\theta)}}(\theta) \\ &> f(\bar{\theta}) \end{aligned}$$

Thus, an optimal solution will not satisfy $AC(\theta')$ if $f{\downarrow_{var(\theta)}}(\theta) < f{\downarrow_{var(\theta')}}(\theta')$.

Otherwise, suppose $f{\downarrow_{var(\theta)}}(\theta) = f{\downarrow_{var(\theta')}}(\theta')$ and $\theta <_{lex} \theta'$. Let $\bar{\theta}' \in \Theta_{AC(\theta')}$ be an optimal solution of P. Since $\theta <_{lex} \theta'$ and $\bar{\theta}\backslash\theta = \bar{\theta}'\backslash\theta'$, $\sigma_m^{-1}(\bar{\theta}') <_{lex} \bar{\theta}'$. That is, if an optimal solution $\bar{\theta}'$ satisfies $\Theta_{AC(\theta')}$, there must be another optimal solution $\sigma_m^{-1}(\bar{\theta}')$ which is lexicographically smaller than $\bar{\theta}'$. By contrapositive, the lexicographically smallest optimal solution cannot satisfy $AC(\theta')$. □

6 Dominance Nogood Generation

Given a length l. Using results in Section 5, we can check if an assignment constraint $AC(\theta)$ dominates another $AC(\theta')$ with the same scope. Once this is established, $\neg AC(\theta')$ can be added to P for dominance breaking. Note that we can generate dominance breaking nogoods for all COPs containing objectives and constraints listed in Section 5.

First we would like to consider the number of such pairs of assignment constraints.

Theorem 13. *Given a COP $P = (X, D, C, f)$ and a length $l \leq |X|$. Suppose $max(|D(x_i)|) = d$ for $x_i \in X$. There are $O(\binom{|X|}{l} \cdot \binom{d^l}{2})$ pairs of assignment constraints $AC(\theta)$ and $AC(\theta')$ where $\theta \neq \theta'$ and $var(\theta) = var(\theta') = F$, $|F| = l$.*

Proof. The candidate pairs can be enumerated by first selecting l variables $x_{i_1}, \ldots, x_{i_l}$ from X. And then we select two distinct tuples from $D(x_{i_1}) \times \cdots \times D(x_{i_l})$, which has $\binom{\prod_{k=1,\ldots,l}^{l} |D(x_{i_k})|}{2} \leq \binom{d^l}{2}$ ways in total. Hence, there are totally $O(\binom{|X|}{l} \cdot \binom{d^l}{2})$ such candidate pairs. □

Using the simple generate-and-test method to find and compare all possible pairs of partial assignments is inefficient. We note that sufficient conditions for betterment in Theorems 5 and 6, and those for implied satisfaction in Theorems 7 to 11 are nothing but constraints on the desired pairs of partial assignments. Thus, we model dominance nogood generation as constraint satisfaction problems.

```
int: n; int: l; int: d;
array [1..l] of var 1..n: F;
array [1..l] of var 1..d: v1;
array [1..l] of var 1..d: v2;
constraint increasing(F);

% betterment
...
% implied satisfaction
...
% compatibility
...
```

The above code template shows our basic nogood generation model in MiniZinc [Nethercote *et al.*, 2007]. Given a COP $P = (X, D, C, f)$. We assume that $|X| = n$ and $d = max(|D(x_i)|)$ for $x_i \in X$. Note that the desired pairs of partial assignments θ and θ' have the same scope. We use an array F to represent the common index set of variables in the scope. We also use $v1$ and $v2$ to represent the assigned values in θ and θ' respectively. Thus, if $\exists i \in \{1, \ldots, l\}$ such

that $F[i] = k$, $v1[i] = t_1$ and $v2[i] = t_2$, then $(x_k = t_1) \in \theta$ and $(x_k = t_2) \in \theta'$. Note that there are variable symmetries in the array F, and we break the symmetries by enforcing $F[i] < F[i+1]$ for all $i = 1, \ldots, l-1$.

With the stated variables, one can post constraints for betterment, implied satisfaction and compatibility according to the objective f and constraints $c \in C$. Using Theorems 4 to 12, each constraint/objective of the problem model P corresponds to $O(1)$ constraints in the generation model in MiniZinc. Such a model can be generated mechanically by analyzing the problem model in only one pass in negligible time as compared to nogood generation and model solving time.

7 Experimental Evaluation

In this section, we present experimental results to demonstrate empirically the effectiveness of the generated dominance nogoods. We use MiniZinc 2.2.3 [Nethercote *et al.*, 2007] to model both the problems and nogood generation respectively, and the back-end solver is Chuffed [Ohrimenko *et al.*, 2009][1]. Our experiments use four benchmarks, 10 instances for each problem configuration.

- The 0-1 *knapsack problem (KP)* is a classical combinatorial optimization problem with a *linear objective* and a *linear inequality constraint*. We use instances from https://people.eng.unimelb.edu.au/pstuckey/dom-jump/ where the number of items $n = 100, 150, 200, 250, 300$.
- The *Disjunctively constrained knapsack problem (DCKP)* [Yamada *et al.*, 2002] extends *KP* so that some pairs of items cannot be selected simultaneously. The extra condition can be modeled as a *Boolean disjunction*. For each instance of *KP* with n items, we augment the instance by randomly picking $\lfloor \eta n(n-1)/2 \rfloor$ incompatible pairs of items where $\eta = 0.002$.
- The *Capacitated Concert Hall Scheduling Problem (CHSP)* [Gange and Stuckey, 2018] is to schedule a set A of applications to a set H of concert halls. Each application $a \in A$ has a period $[s_a, e_a]$, an offered price p_a and a requirement r_a, and each concert hall $h \in H$ has a capacity c_h. The problem is to maximize a *separable* function with *domain* and *alldifferent_except_0* constraints. For $n = 20, 25, 30, 35, 40$, we generate random instances with 10 halls and n applications, $1 \leq s_a \leq e_a \leq 100$, $200 \leq r_a, c_h \leq 1000$ and $10 \leq \frac{p_a}{e_a - s_a + 1}$. To demonstrate the effectiveness of our methods on harder instances, we present only instances which cannot be solved by the model with manual dominance breaking constraints within 60 seconds.
- The *Weighted Maximum Cut Problem (WMCP)* is a *submodular* maximization problem to find a partition $(S, \bar{S})$ of vertices in a weighted undirected graph. For $n \in \{35, 40, 45, 50\}$, we generate random graphs with n vertices by independently sampling each edge with probability $p = 0.1$ whose weights are random integers in $[1, 10]$. As far as we know, *no manual dominance breaking constraints* exist for this problem, and we omit such a model in the experiments.

The problem models, the manual dominance breaking constraints, and the search strategies are from the literature.

We compare our method against the basic problem model (**no-dom**), and the model with manual dominance breaking constraints [Chu and Stuckey, 2012; Gange and Stuckey, 2018] (**manual**). We attempt to generate and augment the basic models with all nogoods of length up to 2 (**2-dom**), 3 (**3-dom**) and 4 (**4-dom**), but subjecting to a uniform timeout limit of 3600s for all benchmark instances.

Figures 1(a) to 1(d) show the average solving (blue solid bar) and dominance nogood generation time (red diagonal hatch bar) in *log scale*. The red bars (generation time) is stacked on the blue ones (solving time). Note that the ratio of the bars does not reflect time percentage. The generation time should not be read directly as the length of the red bars, but should be the difference of the total time and the solving time.

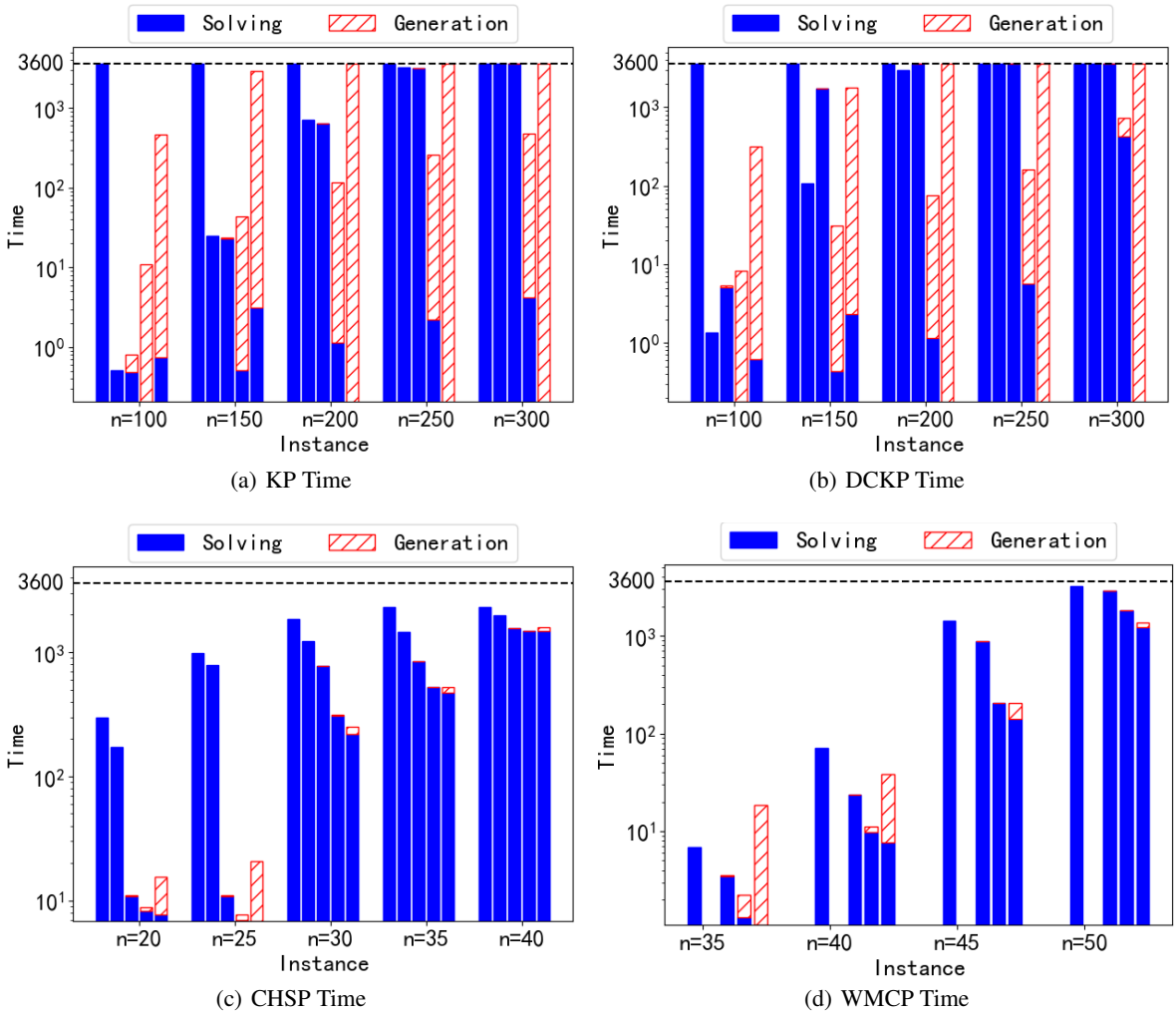
We first compare the problem solving time of our method against **no-dom** and **manual** to understand the strength of our generated dominance breaking nogoods. The five bars for each problem configuration correspond to the time for **manual**, **no-dom**, **2-dom**, **3-dom** and **4-dom** respectively (except for *WMCP* where we omit **manual**). It is clear that the generated dominance breaking constraints can drastically reduce the solving time for all benchmarks. In *KP* and *DCKP*, **no-dom** timeouts in all testing instances, while our method can reduce the solving time to within 10 minutes even for the largest instances. If we choose to compare against the timeout limit, then our method is 17142.6 and 19999.6 times faster than **no-dom** respectively. The speed-up of our method over **no-dom** is up to 125.7 times for *CHSP*, and up to 10.0 times for *WMCP*.

Comparing against **manual**, our method is always more efficient. In particular, our method is up to 1475.0 times faster than that of **manual** for *KP*, and the improvements for *DCKP* and *CHSP* are up to 2568.5 times and 115.4 times respectively. In general, **3-dom** is always better than **2-dom**. When the nogoods of **4-dom** cannot be completely generated within the timeout limit, the comparison of the solving efficiency between **3-dom** and **4-dom** is inconclusive.

We also compare the total time (solving time + generation time) of our method with the solving time of **no-dom** and **manual**. Obviously, more time is needed to generate more nogoods. There is a trade-off between between stronger pruning and generation time. Still, our method comes out on top with **3-dom** being the best. For *KP*, the speed-up is up to 4444.4 times compared with **no-dom** (timeout limit), and 12.5 times compared with **manual**. Similarly for *DCKP*, our method is faster than **no-dom** and **manual** by up to 665.4 times and 38.1 times respectively. As for *CHSP*, our method runs up to 142.2 and 102.0 times faster than both **no-dom** and **manual** respectively. For *WMCP*, our method is up to 7.0 times faster than **no-dom**.

We note that the solving time for many instances exceed the timeout limit, especially for **no-dom** and **manual**, and the

[1] All models and experimental data are available at https://github.com/AllenZzw/Automatic-Dominance-Breaking.



(a) KP Time (b) DCKP Time

(c) CHSP Time (d) WMCP Time

Figure 1: Experimental Results for Comparison: 1(a) to 1(d) compare the nogood generation and problem solving time in log scale.

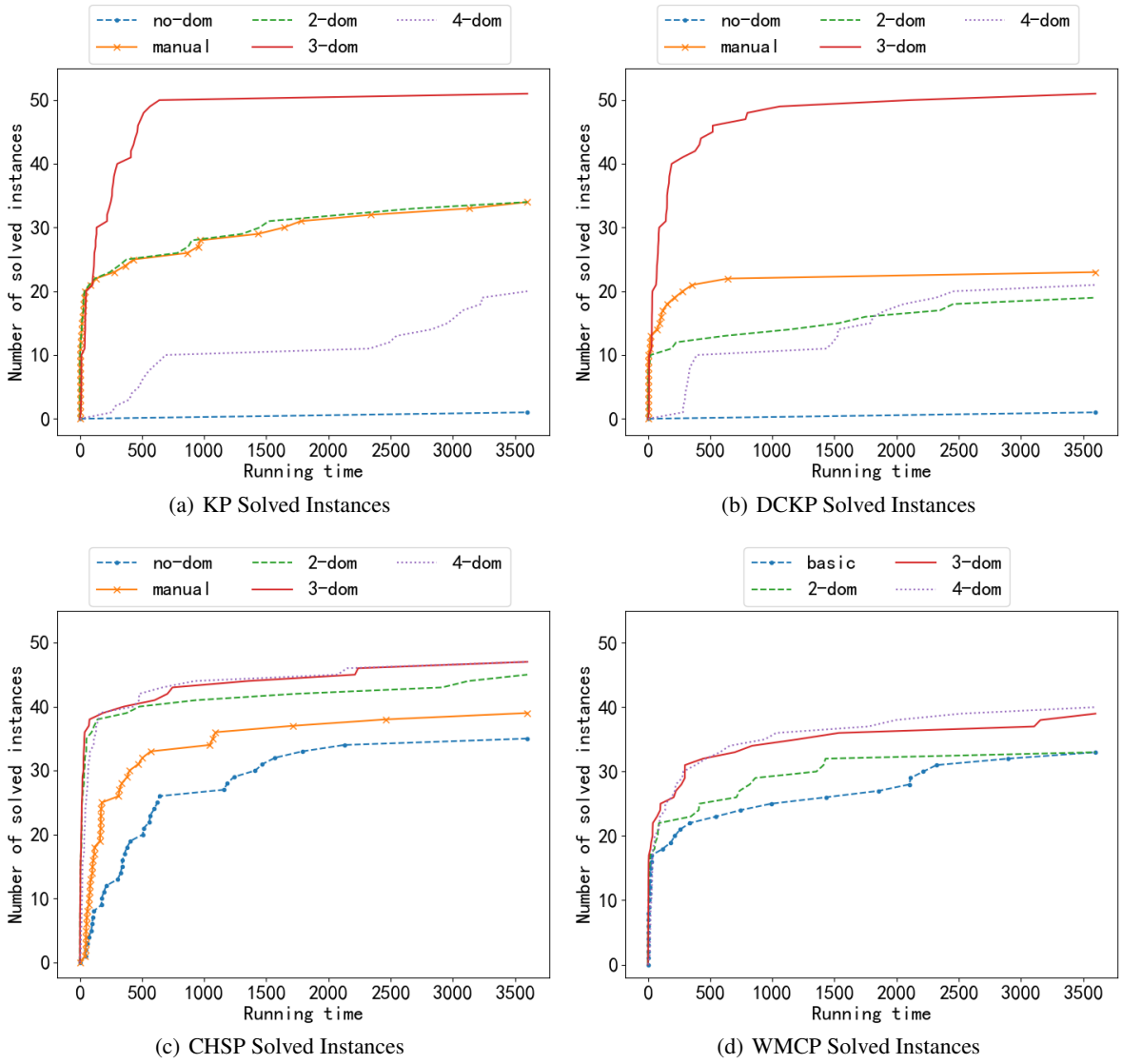
actual acceleration of generated nogoods can be even higher.

Figures 2(a) to 2(d) compare different methods by showing the number of solved instances versus the running time for each benchmark. For both *KP* and *DCKP*, **3-dom** can solve all instances within 10 minutes and 50 minutes respectively, and is better than all other methods, while **no-dom** times out on all instances. In addition, **manual** solves only around 30 and 20 instances out of 50 for *KP* and *DCKP* respectively. For *CHSP*, **no-dom** and **manual** can solve at most 38 instances out of 50, while all these instances can be solved by our method within 10 minutes. As for WCMP, the performance of **4-dom** is the best among all solving methods. **no-dom** can only solve 32 instances out of 40, while **4-dom** can solve 39.

While we do not show pruning strength in the graphs, **3-dom** and **4-dom** all prune substantially more than **manual**. In other words, our method is able to generate dominance breaking nogoods that have not been discovered by manual derivation methods.

8 Concluding Remarks

Automatic generation of dominance breaking constraints is made possible by focusing on nogoods. Our theorems on sufficient conditions enable us to formulate nogood generation effectively as constraint satisfaction. An important advantage is the ability to control the strength of the generated nogoods. Our method discovers dominance breaking nogoods that had not been discovered before. The method can also be easily integrated into existing constraint modeling systems or solvers. There are several future research directions. A limitation of our method is that the nogoods are generated instance by instance, and the benefit of the dominance nogoods may not compensate the overhead for generation. One possibility is to improve the efficiency of nogood generation in general. Another limitaion is that our method can only be applied when the objective and all constraints in the problem can be analysed for betterment and implied satisfaction. We also expect similar results to be proven for other objectives and constraints, especially for global constraints with some



(a) KP Solved Instances

(b) DCKP Solved Instances

(c) CHSP Solved Instances

(d) WMCP Solved Instances

Figure 2: Experimental Results for Comparison: 2(a) to 2(d) show the number of solved instances versus total time.

sort of monotonic behavior.

Acknowledgments

We are grateful to the anonymous referees of IJCAI-20 for their useful comments and suggestions.

References

[Aldowaisan, 2001] Tariq Aldowaisan. A new heuristic and dominance relations for no-wait flowshops with setups. *Computers & Operations Research*, 28(6):563–584, 2001.

[Beldiceanu *et al.*, 2010] Nicolas Beldiceanu, Mats Carlsson, and Jean-Xavier Rampon. Global constraint catalog, 2010.

[Chu and Stuckey, 2012] Geoffrey Chu and Peter J Stuckey. A generic method for identifying and exploiting dominance relations. In *Principles and Practice of Constraint Programming*, pages 6–22, 2012.

[Gange and Stuckey, 2018] Graeme Gange and Peter J Stuckey. Sequential precede chain for value symmetry elimination. In *International Conference on Principles and Practice of Constraint Programming*, pages 144–159. Springer, 2018.

[Getoor *et al.*, 1997] Lise Getoor, Greger Ottosson, Markus Fromherz, and Björn Carlson. Effective redundant constraints for online scheduling. In *The Eleventh AAAI Conference on Artificial Intelligence*, pages 302–307, 1997.

[Ibaraki, 1977] Toshihide Ibaraki. The power of dominance relations in branch-and-bound algorithms. *Journal of the ACM (JACM)*, 24(2):264–279, 1977.

[Katsirelos and Bacchus, 2005] George Katsirelos and Fahiem Bacchus. Generalized nogoods in CSPs. In *The Nineteenth AAAI Conference on Artificial Intelligence*, pages 390–396, 2005.

[Korf, 2004] Richard E Korf. Optimal rectangle packing: New results. In *The Fourteenth International Conference on Automated Planning and Scheduling*, pages 142–149, 2004.

[Land and Doig, 1960] AH Land and AG Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[Mears and de la Banda, 2015] Christopher Mears and Maria Garcia de la Banda. Towards automatic dominance breaking for constraint optimization problems. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[Monette *et al.*, 2007] Jean-Noël Monette, Pierre Schaus, Stéphane Zampelli, Yves Deville, Pierre Dupont, et al. A cp approach to the balanced academic curriculum problem. In *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, volume 7, 2007.

[Nethercote *et al.*, 2007] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.

[Ohrimenko *et al.*, 2009] Olga Ohrimenko, Peter J Stuckey, and Michael Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, 2009.

[Prestwich and Beck, 2004] Steven Prestwich and J Christopher Beck. Exploiting dominance in three symmetric problems. In *Fourth international workshop on symmetry and constraint satisfaction problems*, pages 63–70, 2004.

[Régin, 1994] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *The Eighth AAAI Conference on Artificial Intelligence*, pages 362–367, 1994.

[Yamada *et al.*, 2002] Takeo Yamada, Seija Kataoka, and Kohtaro Watanabe. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal*, 43(9), 2002.