Towards More Practical and Efficient Automatic Dominance Breaking

Jimmy H.M. Lee and Allen Z. Zhong[1]
[1] **Department of Computer Science and Engineering**
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{jlee, zwzhong}@cse.cuhk.edu.hk

Abstract

Dominance breaking is shown to be an effective technique to improve the solving speed of Constraint Optimization Problems (COPs). The paper proposes separate techniques to generalize and make more efficient the nogood generation phase of an automated dominance breaking framework by Lee and Zhong's. The first contribution is in giving conditions that allow skipping the checking of non-efficiently checkable constraints and yet still produce sufficient useful nogoods, thus opening up possibilities to apply the technique on COPs that were previously impractical. The second contribution identifies and avoids the generation of dominance breaking nogoods that are both logically and propagation redundant. The nogood generation model is strengthened using the notion of Common Assignment Elimination to avoid generation of nogoods that are subsumed by other nogoods, thus reducing the search space substantially. Extensive experimentation confirms the benefits of the new proposals.

Introduction

Dominance relations in Constraint Optimization Problems (COPs) describe relations between two full assignments where one is known to be subordinate compared with another with respect to satisfiability and/or objective value. Such relations, if and when discovered, can be used to speed up the Branch and Bound solving process by reducing the search space significantly. A wealth of research investigates the exploitation of dominance relations in COPs, most of which focuses on problem-specific dominance relations and usually requires sophisticated insights of the problem structure (Getoor et al. 1997; Aldowaisan 2001; Korf 2004; Prestwich and Beck 2004; Monette et al. 2007). Chu and Stuckey (2012) presents a generic method for identifying and exploiting dominance relations, and it is automated to a large extent by Mears and Garcia de la Banda (2015) which still calls for human interventions in the selection of symmetries to generate effective dominance breaking constraints.

Lee and Zhong (2020) give the theories and techniques that allow the mechanical construction of a CSP model to generate dominance breaking nogoods for a useful class of COPs, which can then be added to the problem model to speed up solving. The framework is able to identify dominance breaking nogoods that had not been discovered before but are stronger than the manually constructed constraints.

In this paper, we extend the applicability of Lee and Zhong's work with two theoretical and practical innovations. First, the original Lee and Zhong method requires all constraints to be efficiently checkable (EC) to guarantee the efficiency of the nogood generation process. We prove formally on when and how non-EC constraints can be ignored in nogood generation by not exploring nogoods that contain variables in the non-EC constraints. This way, sufficient useful nogoods can still be generated when the number of variables in non-EC constraints are relatively small compared to all variables of the problem. Second, we show that some generated nogoods make no contributions in pruning since they are both logically and propagation redundant (Choi, Lee, and Stuckey 2003) with respect to other nogoods. We propose Common Assignment Elimination to ban generation of such fruitless nogoods, thus speeding up the generation process substantially. Experimentation confirms the enhanced applicability of our theory-backed methods, which allow us to tackle benchmarks that could not be handled by the original Lee and Zhong method.

Background

Constraint Satisfaction and Optimization

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 $var(c) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$. A *Constraint Optimization 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 *on variables* X' is a set of assignments, one for each variable in $X' \subset X$, i.e. $\theta = \{x_{i_j} = v_{i_j} \mid x_{i_j} \in X' \wedge \forall i_j \neq i_{j'}, x_{i_j} \neq x_{i_{j'}}\}$, where the set $X' = var(\theta)$ is the *scope* of θ. 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 $|X'| = l$. Note that a partial assignment can also be interpreted as a conjunction of equational constraints, and a *nogood* $\neg\theta$ is simply the negation of the conjunction for θ.

A *full assignment* is a partial assignment when $X' = X$, and we use $\bar{\theta}$ to emphasize that it is a full assignment. The *projection* $\theta{\downarrow_Y}$ *of* θ *onto* $Y \subseteq var(\theta)$ is the partial assignment $\{x = v \mid (x = v) \in \theta \wedge x \in Y\}$. 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. If the set of all solutions $sol(P)$ is non-empty, then P is *satisfiable*.

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.

Dominance Relations

Let Θ^P be the set of all full assignments of a COP $P = (X, D, C, f)$. A *dominance relation* $\prec$ *over* Θ^P (Chu and Stuckey 2012) is a transitive and irreflexive relation such that for $\bar{\theta}, \bar{\theta}' \in \Theta^P$, if $\bar{\theta}$ *dominates* $\bar{\theta}'$, i.e. $\bar{\theta} \prec \bar{\theta}'$, with respect to P, then either: (1) $\bar{\theta} \in sol(P)$ and $\bar{\theta}' \notin sol(P)$, or (2) $\bar{\theta}, \bar{\theta}' \in sol(P)$ and $f(\bar{\theta}) \leq f(\bar{\theta}')$, or (3) $\bar{\theta}, \bar{\theta}' \notin sol(P)$ and $f(\bar{\theta}) \leq f(\bar{\theta}')$. Dominance relations can be generalized to partial assignments. Let $\Theta^P_\theta = \{\bar{\theta} \in \Theta^P \mid \bar{\theta} \supseteq \theta\}$ be the subset of full assignments that satisfy θ. By abusing notation, we say that a partial assignment θ *dominates* another θ', i.e. $\theta \prec \theta'$, with respect to P if and only if $\forall \bar{\theta}' \in \Theta^P_{\theta'}, \exists \bar{\theta} \in \Theta^P_\theta$ such that $\bar{\theta} \prec \bar{\theta}'$ with respect to P (Lee and Zhong 2020). The following theorem states that if $\theta \prec \theta'$ with respect to P, it is sound to prune all assignments in $\Theta^P_{\theta'}$ when searching for P's optimal solutions.

Theorem 1. *(Lee and Zhong 2020) Suppose that* θ *and* θ' *are partial assignments of* $P = (X, D, C, f)$ *such that* $var(\theta) = var(\theta')$. *If* $\theta \prec \theta'$ *with respect to* P, *then* P *has the same satisfiability or optimal value as* $P \cup \neg\theta'$.

Theorem 1 implies that if θ' is dominated by some partial assignment (θ), then $P \cup \neg\theta'$ and P have the same satisfiability or optimal value. Furthermore, $P \cup \neg\theta'$ has a smaller search space, and the constraint $\neg\theta'$ is called a *dominance breaking constraint for* P. Since it is in the form of *nogood*, we also call it a *dominance breaking nogood* for P.

Automatic Dominance Breaking

Lee and Zhong (2020) show how to automate the identification and exploitation of dominance relations as follows:

1. The objective and constraints of a COP P are analyzed and a (dominance breaking nogood) *generation problem* P_g as a CSP is constructed mechanically. Solutions of P_g are not nogoods, but can be used to create nogoods.
2. All solutions of P_g are enumerated, and each solution corresponds to a pair of partial assignments (θ, θ') such that $\theta \prec \theta'$ with respect to P.
3. The corresponding dominance breaking nogoods $\{\neg\theta'\}$ are collected and added to P before solving.

The key step for automation is to construct P_g mechanically from P, and yet all solutions of P_g can be generated efficiently. Note that the difficulty of an all-solution search for a CSP depends only on the number of variables and the respective domain size in P_g. Thus, having more constraints in P_g would only speed up solving since there will be less solutions and more pruning. In case P_g is unsatisfiable, it only means that there are no dominance breaking nogoods.

By definition, one can directly check every full assignment in $\Theta^P_{\theta'}$ against all full assignments in Θ^P_θ to establish $\theta \prec \theta'$, but it is impractical. To improve efficiency, P_g only focuses on searching for (θ, θ') such that $var(\theta) = var(\theta')$, and a bijective mapping called *mutation mapping* $\mu^{\theta\to\theta'} : \Theta^P_\theta \to \Theta^P_{\theta'}$ is further introduced. A full assignment $\bar{\theta} \in \Theta^P_\theta$ is mapped to $\mu^{\theta\to\theta'}(\bar{\theta})$ such that $\mu^{\theta\to\theta'}(\bar{\theta}) = (\bar{\theta} \setminus \theta) \cup \theta'$. Solutions of P_g will correspond to only pairs $\{(\theta, \theta')\}$ such that $\bar{\theta} \prec \mu^{\theta\to\theta'}(\bar{\theta})$ with respect to P for every $\bar{\theta} \in \Theta^P_\theta$.

Example 1. *Consider a simple COP* P *as follows:*

$$\begin{aligned} &\textit{minimize} && \sum_{i=1}^{3} a_i \cdot x_i \\ &\textit{subject to} && \sum_{i=1}^{3} b_i \cdot x_i \geq 3 \\ & && x_i \in \{0,1\}\ \textit{for}\ i = 1,2,3 \end{aligned}$$

where $a = [1, 4, 2]$ *and* $b = [3, 2, 1]$ *are two arrays of integers. For two partial assignments* $\theta = \{x_1 = 1\}$ *and* $\theta' = \{x_1 = 0\}$, *the sets* Θ^P_θ *and* $\Theta^P_{\theta'}$ *both contain 4 full assignments. Given a full assignment* $\bar{\theta} \in \Theta^P_\theta$, *the mutation mapping* $\mu^{\theta\to\theta'} : \Theta^P_\theta \to \Theta^P_{\theta'}$ *transforms the* θ *component of* $\bar{\theta}$ *to become* $\bar{\theta}' = \mu^{\theta\to\theta'}(\bar{\theta}) \in \Theta^P_{\theta'}$. *We enumerate the mapping in the following table:*

	$\bar{\theta}$				$\bar{\theta}'$	
x_1	x_2	x_3		x_1	x_2	x_3
1	*0*	*0*	$\mapsto$	*0*	*0*	*0*
1	*0*	*1*	$\mapsto$	*0*	*0*	*1*
1	*1*	*0*	$\mapsto$	*0*	*1*	*0*
1	*1*	*1*	$\mapsto$	*0*	*1*	*1*

The transformed parts are highlighted in color.

Checking for $\bar{\theta} \prec \mu^{\theta\to\theta'}(\bar{\theta})$ with respect to P for every $\bar{\theta} \in \Theta^P_\theta$ is still expensive in general, and thus a sufficient condition is given as follows.

Theorem 2. *(Lee and Zhong 2020) Suppose that* θ *and* θ' *are partial assignments of a COP* $P = (X, D, C, f)$ *such that* $var(\theta) = var(\theta')$. *If* (θ, θ') *satisfies:*

- *empty intersection:* $\Theta^P_\theta \cap \Theta^P_{\theta'} = \emptyset$
- *betterment:* $\forall \bar{\theta} \in \Theta^P_\theta$, $f(\bar{\theta}) \leq f(\mu^{\theta\to\theta'}(\bar{\theta}))$
- *implied satisfaction:* $\forall \bar{\theta} \in \Theta^P_\theta$, $\mu^{\theta\to\theta'}(\bar{\theta}) \in sol(P)$ *implies* $\bar{\theta} \in sol(P)$.

then $\theta \prec \theta'$ *with respect to* P.

By Theorem 2, empty intersection, betterment and implied satisfaction together form a sufficient condition, and thus jointly they can prove that $\theta \prec \theta'$ with respect to P. Empty intersection is easy to check: if $\theta \neq \theta'$, then $\Theta^P_\theta \cap$

$\Theta^P_{\theta'} = \emptyset$. However, checking betterment and implied satisfaction requires comparing every $\bar{\theta} \in \Theta^P_\theta$ against $\mu^{\theta\to\theta'}(\bar{\theta})$, which is still costly in general. Lee and Zhong (2020) further give sufficient conditions on (θ, θ') for certain classes of objectives and constraints in P so that betterment and implied satisfaction can be checked efficiently. We say that objective and constraints with such sufficient conditions are *efficiently checkable* (EC) and these sufficient conditions can be modelled as constraints in P_g. As long as the objective and all constraints of the COP P are EC, we can automatically construct efficiently solvable P_g. We summarize the constraints in P_g in the following definition.

Definition 1 (The Generation Problem). *Each pair (θ, θ') of partial assignments resulting from solutions of the* generation problem P_g *should fulfil: (1) same scope: $var(\theta) = var(\theta')$, (2) unequal pair: $\theta \neq \theta'$, (3) sufficient conditions on (θ, θ') for betterment arising from the objective of P, and (4) sufficient conditions on (θ, θ') for implied satisfaction arising from the constraints of P.*

Note that the above are high-level description of constraints in P_g. In practice, we usually limit the maximum size $|var(\theta)|$ to be a fixed L, which is the length of the maximum length of generated dominance breaking nogoods. The actual implementation uses three arrays of size at most L to represent the indices of variables and the assigned values in θ and θ' respectively (Lee and Zhong 2020).

Example 2. *Consider searching for pairs (θ, θ') such that $\theta \prec \theta'$ with respect to the COP P in Example 1. The problem P has a linear objective function and a linear inequality constraint. Let $\sum_i a_i x_i$ be a linear expression, and $(\sum_i a_i x_i)\theta$ be the expression by replacing every occurrence of variable x_i by value v_i for all assignment $(x_i = v_i) \in \theta$. Following Lee and Zhong (2020), the sufficient conditions are as follows:*

- *Sufficient conditions for betterment:*

$$\sum_{i=1}^{3}(a_i \cdot x_i)\theta < \sum_{i=1}^{3}(a_i \cdot x_i)\theta' \equiv \sum_{(x_i=v_i)\in\theta} a_i \cdot v_i < \sum_{(x_i=v'_i)\in\theta'} a_i \cdot v'_i$$

- *Sufficient conditions for implied satisfaction:*

$$\sum_{i=1}^{3}(b_i \cdot x_i)\theta \geq \sum_{i=1}^{3}(b_i \cdot x_i)\theta' \equiv \sum_{(x_i=v_i)\in\theta} b_i \cdot v_i \geq \sum_{(x_i=v'_i)\in\theta'} b_i \cdot v'_i$$

By Theorem 2, if a pair (θ, θ') of partial assignments fulfils same scope, unequal pair and the above sufficient conditions, then $\theta \prec \theta'$ with respect to P. One such pair is $\theta = \{x_1 = 1, x_2 = 0\}$ and $\theta' = \{x_1 = 0, x_2 = 1\}$, since

$$var(\theta) = \{x_1, x_2\} = var(\theta'),$$

$$\theta = \{x_1 = 1, x_2 = 0\} \neq \{x_1 = 0, x_2 = 1\} = \theta',$$

$$\sum_{(x_i=v_i)\in\theta} a_i \cdot v_i = 1 + 0 < 0 + 4 = \sum_{(x_i=v'_i)\in\theta'} a_i \cdot v'_i,$$

$$\sum_{(x_i=v_i)\in\theta} b_i \cdot v_i = 3 + 0 \geq 0 + 2 = \sum_{(x_i=v'_i)\in\theta'} b_i \cdot v'_i$$

By Theorem 1, $\neg(x_1 = 0 \wedge x_2 = 1)$ is a dominance breaking nogood for P in Example 1.

Non-Efficiently Checkable Constraints

By Theorem 2, the implied satisfaction for a pair (θ, θ') of partial assignments requires that $\forall\bar{\theta} \in \Theta^P_\theta$, $\mu^{\theta\to\theta'}(\bar{\theta}) \in sol(P) \Rightarrow \bar{\theta} \in sol(P)$, and recall that $\bar{\theta} \in sol(P)$ if and only if $\bar{\theta}$ satisfies all constraints $c \in C$. The practicality of automatic dominance breaking requires every constraint $c \in C$ to be EC, which means there are sufficient conditions on (θ, θ') to show that $\bar{\theta}$ satisfies c when $\mu^{\theta\to\theta'}(\bar{\theta})$ satisfies c. This is, however, not always possible. Lee and Zhong (2020) only give such sufficient conditions for certain classes of constraints. What if P contains constraints with no known sufficient conditions for implied satisfaction (yet)? We propose a way to allow skipping the checking of such non-EC constraints and yet some dominance breaking nogoods can still be found.

We first review a useful proposition for showing implied satisfaction. Let $c\theta$ be the constraint obtained by replacing every occurrence of variable x in c by value v for every assignment $(x = v) \in \theta$.

Proposition 1. *(Lee and Zhong 2020) Suppose that $c \in C$ is a constraint of a COP $P = (X, D, C, f)$. If $c\theta' \Rightarrow c\theta$, then $\mu^{\theta\to\theta'}(\bar{\theta}) \in c \Rightarrow \bar{\theta} \in c, \forall\bar{\theta} \in \Theta^P_\theta$.*

For EC constraints, there are sufficient conditions to show that $c\theta' \Rightarrow c\theta$. To skip checking for non-EC constraints, we rely on the following useful property.

Lemma 1. *Suppose that θ and θ' are partial assignments of a COP $P = (X, D, C, f)$ such that $var(\theta) = var(\theta')$, and $c \in C$ is a constraint. If $var(\theta) \cap var(c) = \emptyset$, then $\bar{\theta}\downarrow_{var(c)} = \mu^{\theta\to\theta'}(\bar{\theta})\downarrow_{var(c)}$ for every $\bar{\theta} \in \Theta^P_\theta$.*

Proof. Let $\bar{\theta}' = \mu^{\theta\to\theta'}(\bar{\theta})$. The mutation mapping $\mu^{\theta\to\theta'}$ replaces the θ component of $\bar{\theta} \in \Theta^P_\theta$ with θ', i.e. $\bar{\theta} \setminus \theta = \bar{\theta}' \setminus \theta'$. If $var(c) \cap var(\theta) = \emptyset$, then $\bar{\theta}\downarrow_{var(c)} = (\bar{\theta} \setminus \theta)\downarrow_{var(c)} = (\bar{\theta}' \setminus \theta')\downarrow_{var(c)} = \bar{\theta}'\downarrow_{var(c)} = \mu^{\theta\to\theta'}(\bar{\theta})\downarrow_{var(c)}$. ☐

Recall that a full assignment satisfies a constraint if $\bar{\theta}\downarrow_{var(c)} \in c$. By Lemma 1, when $var(\theta) \cap var(c) = \emptyset$, $\bar{\theta}\downarrow_{var(c)} \in c \Leftrightarrow \mu^{\theta\to\theta'}(\bar{\theta})\downarrow_{var(c)} \in c$ for all $\bar{\theta} \in \Theta^P_\theta$. Thus, the pair (θ, θ') trivially satisfies implied satisfaction, and do not have to be checked.

Theorem 3. *Suppose that θ and θ' are two partial assignments of a COP $P = (X, D, C_1 \cup C_2, f)$ where $F = var(\theta) = var(\theta') \subseteq X$, and $\mu^{\theta\to\theta'}$ is the associated mutation mapping. If (θ, θ') fulfils that: (1) for all $c \in C_1$, $c\theta' \Rightarrow c\theta$, and (2) for all $c \in C_2$, $F \cap var(c) = \emptyset$, then $\forall\bar{\theta} \in \Theta^P_\theta$, $\mu^{\theta\to\theta'}(\bar{\theta}) \in sol(P) \Rightarrow \bar{\theta} \in sol(P)$.*

Proof. Let $\bar{\theta} \in \Theta^P_\theta$. (1) Suppose that $c \in C_1$. By Proposition 1, since $c\theta' \Rightarrow c\theta$, $\mu^{\theta\to\theta'}(\bar{\theta}) \in c \Rightarrow \bar{\theta} \in c$. (2) Otherwise, $c \in C_2$. Implied satisfaction hold automatically by Lemma 1. Thus, $\bar{\theta}$ satisfies all constraints in P if $\mu^{\theta\to\theta'}(\bar{\theta})$ satisfies all constraints in P, which means that $\mu^{\theta\to\theta'}(\bar{\theta}) \in sol(P) \Rightarrow \bar{\theta} \in sol(P)$. ☐

In general, if a constraint c of P is non-EC, we can add an extra condition $var(\theta) \cap var(c) = \emptyset$ in P_g so that θ containing $var(c)$ would not be explored for identifying dominance breaking nogoods. By Theorems 2 and 3, we can still find some pairs (θ, θ') such that $\theta \prec \theta'$ with respect to P, but the generated nogoods will not involve variables in c. In other words, this method is useful only if $var(c)$ is relatively small with respect to X. Otherwise, very few nogoods can be generated.

Common Assignment Elimination

Let P_g be a generation problem arising from a COP P, and $S(P_g)$ be the set of (θ, θ') obtained by solving P_g, i.e. $(\theta, \theta') \in S(P_g)$ if and only if all (θ, θ') fulfils conditions in Definition 1. Thus, there is a one-one correspondence between $sol(P_g)$ and $S(P_g)$. We augment P with all dominance breaking nogoods $\neg\theta'$ resulting from $(\theta, \theta') \in S(P_g)$. In a propagation solver, a nogood constraint $c \equiv \neg\theta'$ is usually enforced to be *generalized arc consistent (GAC)* (Mackworth and Freuder 1985), where for every $x_{i_j} \in var(c)$ and $v \in D(x_{i_j})$, there exists a tuple $(v_{i_1}, \ldots, v_{i_{j-1}}, v, v_{i_{j+1}}, \ldots, v_{i_k}) \in c$, and we call the tuple a *support* for value v. If the constraint is already GAC, constraint propagation will not remove any further values from the domain (Schulte and Stuckey 2008).

Proposition 2. *Suppose that θ' and $\tilde{\theta}'$ are partial assignments of a COP $P = (X, D, C, f)$. If $\tilde{\theta}' \subset \theta'$, then (1) $\neg\tilde{\theta}' \Rightarrow \neg\theta'$, and (2) $\neg\tilde{\theta}'$ is GAC $\Rightarrow \neg\theta'$ is GAC.*

Proposition 2 is easy to check and implies that if $\tilde{\theta}' \subset \theta'$, then $\neg\theta'$ is both logically and propagation redundant with respect to $\neg\tilde{\theta}'$. Note that a propagation redundant constraint does not contribute additional information to the constraint solver (Choi, Lee, and Stuckey 2003).

Recall that a dominance breaking nogood arises from a pair $(\theta, \theta') \in S(P_g)$. If there exists another pair $(\tilde{\theta}, \tilde{\theta}') \in S(P_g)$ such that $\tilde{\theta}' \subset \theta'$, then $\neg\tilde{\theta}'$ is added to P. There is no need to generate (θ, θ') and augment the model of P with $\neg\theta'$ since $\neg\theta'$ is logically and propagation redundant with respect to $\neg\tilde{\theta}'$ by Proposition 2. Thus, it suffices to compute only the set $\{(\theta, \theta') \in S(P_g) \mid \forall(\tilde{\theta}, \tilde{\theta}') \in S(P_g), \tilde{\theta}' \not\subset \theta'\}$.

We propose to further enhance the generation problem P_g with extra conditions of (θ, θ') to avoid generating a pair (θ, θ') of partial assignments such that $\neg\theta$ is propagation redundant with respect to all other generated nogoods. In particular, we are interested in the case when θ and θ' share a common assignment $(x = v) \in \theta \cap \theta'$.

Definition 2 (Eliminability of an Assignment). *An assignment $(x = v)$ is* commonly eliminable with respect to P_g *if and only if* $\forall(\theta, \theta') \in S(P_g), (x = v) \in \theta \cap \theta' \Rightarrow (\tilde{\theta}, \tilde{\theta}') \in S(P_g)$ *where* $\tilde{\theta} = \theta \setminus \{x = v\}$ *and* $\tilde{\theta}' = \theta' \setminus \{x = v\}$.

Immediately, we have the following theorem.

Theorem 4. *Let P_g be a generation problem resulting from a COP $P = (X, D, C, f)$. If an assignment $(x = v)$ is commonly eliminable with respect to P_g, then $\forall(\theta, \theta') \in S(P_g), \exists(\tilde{\theta}, \tilde{\theta}') \in S(P_g)$ such that $(x = v) \notin \tilde{\theta} \cap \tilde{\theta}'$ and $\tilde{\theta}' \subseteq \theta'$.*

Proof. Suppose that $(x = v)$ is not a common assignment of (θ, θ'), i.e. $(x = v) \notin \theta \cap \theta'$. The theorem holds by letting $\tilde{\theta} = \theta$ and $\tilde{\theta}' = \theta'$.

Otherwise, $(x = v) \in \theta \cap \theta'$. Let $\tilde{\theta} = \theta \setminus \{x = v\}$ and $\tilde{\theta}' = \theta' \setminus \{x = v\}$. Immediately, we have $\tilde{\theta}' \subset \theta'$ and $(x = v) \notin \tilde{\theta}' \cap \tilde{\theta}$. Since $(x = v)$ is commonly eliminable with respect P_g, $(\theta, \theta') \in S(P_g)$ implies that $(\tilde{\theta}, \tilde{\theta}') \in S(P_g)$. □

Theorem 4 states that if an assignment $(x = v) \in \theta \cap \theta'$ is commonly eliminable, then there must exist $(\tilde{\theta}, \tilde{\theta}')$ of smaller size, and $\neg\theta'$ is logically and propagation redundant with respect to $\neg\tilde{\theta}'$ by Proposition 2. So we do not want (θ, θ') to be generated when solving P_g, and we can add the constraint $(x = v) \notin \theta \cap \theta'$ into P_g. The technique is called *Common Assignment Elimination (CAE)*. By Proposition 2, the collective strength of the resulting dominance breaking nogoods in pruning search space for P after applying CAE remains unchanged. Thus, the key question here is how to show that an assignment is commonly eliminable.

By definition, checking eliminability of an assignment with respect to P_g requires us to go through each of the four conditions in Definition 1. It is straightforward to show that *any arbitrary assignment* $(x = v)$ fulfils the eliminability conditions for same scope and unequal pair.

Proposition 3. *Suppose θ, θ' are partial assignments for a COP P. If $(x = v) \in \theta \cap \theta'$, then (1) $(var(\theta) = var(\theta')) \Rightarrow (var(\tilde{\theta}) = var(\tilde{\theta}'))$, and (2) $(\theta \neq \theta') \Rightarrow (\tilde{\theta} \neq \tilde{\theta}')$, where $(\tilde{\theta}, \tilde{\theta}') = (\theta \setminus \{x = v\}, \theta' \setminus \{x = v\})$.*

What remains is to analyze what kind of assignments fulfil the eliminability conditions for betterment and implied satisfaction. To do that, we examine and exploit the sufficient conditions for betterment and implied satisfaction given by Lee and Zhong (2020) for specific objectives and constraints respectively. Unless otherwise stated, all formal results in the remainder of this section are given **within the following context**: *suppose θ and θ' are two partial assignments of a COP $P = (X, D, C, f)$, where $var(\theta) = var(\theta')$. The mutation mapping for (θ, θ') is $\mu^{\theta \to \theta'} : \Theta^P_\theta \to \Theta^P_{\theta'}$.*

The Betterment Condition

This subsection reviews sufficient conditions for the betterment condition: if $\forall\bar{\theta} \in \Theta^P_\theta$, then $f(\bar{\theta}) \leq f(\mu^{\theta \to \theta'}(\bar{\theta}))$ where f is the objective of a COP P. To prove that a common assignment $(x = v) \in \theta \cap \theta'$ fulfils the eliminability condition for betterment, we show that the sufficient condition holds for $(\tilde{\theta}, \tilde{\theta}') = (\theta \setminus (x = v), \theta' \setminus (x = v))$ whenever (θ, θ') does. We consider two types of objectives: separable objectives and supermodular/submodular objectives.

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 $\bar{\theta} = \{(x_i = v_i) \mid x_i \in X\}$, $f(\bar{\theta}) = f_1(v_1) + \cdots + f_n(v_n)$, where each component is $f_i : \mathbb{Z} \to \mathbb{R}$. For a partial assignment θ, the *projection* is $f\downarrow_{var(\theta)} (\theta) = f_{i_1}(v_{i_1}) + \cdots + f_{i_l}(v_{i_l})$ where $(x_{i_j} = v_{i_j}) \in \theta$ for $j \in \{1, \ldots, l\}$.

Lemma 2. *(Lee and Zhong 2020) Suppose the objective f is a separable function. If $f{\downarrow_{var(\theta)}}(\theta) \leq f{\downarrow_{var(\theta')}}(\theta')$, then $\forall \bar{\theta} \in \Theta_\theta^P$, $f(\bar{\theta}) \leq f(\mu^{\theta \to \theta'}(\bar{\theta}))$.*

The theorem below states that *an arbitrary assignment* $(x = v)$ fulfils the eliminability condition with respect to betterment if $(x = v) \in \theta \cap \theta'$ and f is separable.

Theorem 5. *Suppose the objective f is a separable function. If $(x_t = v_t) \in \theta \cap \theta'$, then $f{\downarrow_{var(\theta)}}(\theta) \leq f{\downarrow_{var(\theta')}}(\theta')$ implies that $f{\downarrow_{var(\tilde{\theta})}}(\tilde{\theta}) \leq f{\downarrow_{var(\tilde{\theta}')}}(\tilde{\theta}')$ where $\tilde{\theta} = \theta \setminus \{x_t = v_t\}$ and $\tilde{\theta}' = \theta' \setminus \{x_t = v_t\}$.*

Proof. If $f{\downarrow_{var(\theta)}}(\theta) \leq f{\downarrow_{var(\theta')}}(\theta')$, then $f{\downarrow_{var(\tilde{\theta})}}(\tilde{\theta}) = f{\downarrow_{var(\theta)}}(\theta) - f_t(v_t) \leq f{\downarrow_{var(\theta')}}(\theta') - f_t(v_t) = f{\downarrow_{var(\tilde{\theta}')}}(\tilde{\theta}')$. □

Supermodular and Submodular Objectives A *supermodular function* is a set function $F : 2^V \to \mathbb{R}$ that assigns each subset $U \subseteq V$ a value $g(U) \in \mathbb{R}$ such that

$$F(U \cup T) - F(U) \leq F(U' \cup T) - F(U')$$

for every $U, U' \subseteq V$ with $U \subseteq U'$ and $T \subseteq V \setminus U'$. A set function F is *submodular* if $-F$ is supermodular. Given a partial/full assignment θ over 0-1 variables, we define a set $U(\theta) = \{i \mid (x_i = 1) \in \theta\}$. In the following, we say that a function f on 0-1 variables is *equivalent to a supermodular function* F if $f(\bar{\theta}) = F(U(\bar{\theta}))$ for any full assignment $\bar{\theta}$ of P, and consider functions of this kind.

Lemma 3. *(Lee and Zhong 2020) Suppose the objective f is a function on 0-1 variables and is equivalent to a supermodular function F. If $U(\theta) \subseteq U(\theta')$ and $F(U(\theta)) \leq F(U(\theta'))$, then $\forall \bar{\theta} \in \Theta_\theta^P$, $f(\bar{\theta}) \leq f(\mu^{\theta \to \theta'}(\bar{\theta}))$.*

By definition, removing *a common assignment* $(x = 0)$ from θ and θ' does not affect the represented set $U(\theta)$ and $U(\theta')$. Thus, the sufficient condition hold for $(\tilde{\theta}, \tilde{\theta}') = (\theta \setminus (x = v), \theta' \setminus (x = v))$ whenever (θ, θ') does.

Theorem 6. *Suppose the objective f is a function on 0-1 variables and is equivalent to a supermodular function F. If $(x_t = 0) \in \theta \cap \theta'$, then*

1) $F(U(\theta)) \leq F(U(\theta')) \Rightarrow F(U(\tilde{\theta})) \leq F(U(\tilde{\theta}'))$

2) $U(\theta) \subseteq U(\theta') \Rightarrow U(\tilde{\theta}) \subseteq U(\tilde{\theta}')$

where $\tilde{\theta} = \theta \setminus \{x_t = 0\}$ and $\tilde{\theta}' = \theta' \setminus \{x_t = 0\}$.

Proof. Since $U(\theta) = \{i \mid (x_i = 1) \in \theta\}$, $U(\tilde{\theta}) = U(\theta \setminus \{x_t = 0\}) = U(\theta)$. Similarly, $U(\tilde{\theta}') = U(\theta')$.
1) If $F(U(\theta)) \leq F(U(\theta'))$, then $F(U(\tilde{\theta})) = F(U(\theta)) \leq F(U(\theta')) = F(U(\tilde{\theta}'))$. 2) If $U(\theta) \subseteq U(\theta')$, then $U(\tilde{\theta}) = U(\theta) \subseteq U(\theta') = U(\tilde{\theta}')$. □

The Implied Satisfaction Condition

This section reviews sufficient conditions for implied satisfaction: $\forall \bar{\theta} \in \Theta_\theta^P$ and $c \in C$, $\mu^{\theta \to \theta'}(\bar{\theta})$ satisfies $c \Rightarrow \bar{\theta}$ satisfies c. By Proposition 1, it is sufficient to show that $c\theta' \Rightarrow c\theta$. We will show that the sufficient condition

Domain Constraints A domain constraint $x \in U$ is a unary constraint that restricts a variable x to take values from a set $V \subseteq D(x)$.

Lemma 4. *(Lee and Zhong 2020) Suppose we have a constraint $x \in U$ where $V \subseteq D(x)$. If either (1) $x \notin var(\theta)$ or (2) $\exists (x = v) \in \theta$ s.t. $v \in V$, then $c\theta' \Rightarrow c\theta$.*

Any *arbitrary assignment* $(x = v)$ fulfils the eliminability condition for implied satisfaction for domain constraints.

Theorem 7. *Suppose we have a constraint $x_i \in V$ where $V \subseteq D(x_i)$. If $(x_t = v_t) \in \theta \cap \theta'$, then*

$$(x_i \notin var(\theta)) \vee (\exists (x_i = v_i) \in \theta \text{ s.t. } v_i \in V)$$
$$\Rightarrow (x_i \notin var(\tilde{\theta})) \vee (\exists (x_i = v_i) \in \tilde{\theta} \text{ s.t. } v_i \in V)$$

where $\tilde{\theta} = \theta \setminus \{x_t = v_t\}$ and $\tilde{\theta}' = \theta' \setminus \{x_t = v_t\}$.

Proof. To prove $A \vee B \Rightarrow C$, we show that $A \Rightarrow C$ and $B \Rightarrow C$. Since $var(\tilde{\theta}) \subset var(\theta)$, $(x_i \notin var(\theta)) \Rightarrow (x_i \notin var(\tilde{\theta})) \Rightarrow (x_i \notin var(\tilde{\theta})) \vee (\exists (x_i = v_i) \in \tilde{\theta}$ s.t. $v_i \in V)$.
Next, suppose that $\exists (x_i = v_i) \in \theta$ s.t. $v_i \in V$.

- If x_i and x_t are the same, then so must v_i and v_t by definition of a partial assignment. We have $x_i \notin var(\tilde{\theta})$ since $\tilde{\theta} = \theta \setminus \{x_t = v_t\} = \theta \setminus \{x_i = v_i\}$.
- Otherwise, x_i and x_t are different. $(x_i = v_i) \in \theta \Rightarrow ((x_i = v_i) \in (\theta \setminus \{x_t = v_t\})) \Leftrightarrow (x_i = v_i) \in \tilde{\theta}$.

Thus, $(\exists (x_i = v_i) \in \theta$ s.t. $v_i \in V) \Rightarrow (x_i \notin var(\tilde{\theta})) \vee (\exists (x_i = v_i) \in \tilde{\theta}$ s.t. $v_i \in V)$. □

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

Lemma 5. *(Lee and Zhong 2020) Suppose we have a constraint $\sum_i w_i x_i \leq b$. 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 $c\theta' \Rightarrow c\theta$.*

The sufficient condition still holds when an *arbitrary common assignment* $(x = v) \in \theta \cap \theta'$ is removed.

Theorem 8. *Given a constraint $\sum w_i x_i \leq b$, if $(x_t = v_t) \in \theta \cap \theta'$, then $\sum_{(x_i = v_i) \in \theta} w_i v_i \leq \sum_{(x_i = v_i') \in \theta'} w_i v_i' \Rightarrow \sum_{(x_i = v_i) \in \tilde{\theta}} w_i v_i \leq \sum_{(x_i = v_i') \in \tilde{\theta}'} w_i v_i'$, where $\tilde{\theta} = \theta \setminus \{x_t = v_t\}$ and $\tilde{\theta}' = \theta' \setminus \{x_t = v_t\}$.*

The proof idea is similar to that of Theorem 5.

Boolean Disjunctions The *Boolean disjunction* constraint $\vee_{x \in S} x$ requires that at least one Boolean variable $x \in S$ takes the true value. Here we abuse the notation to consider false as 0 and true as 1. The sufficient condition requires that if θ' assigns 1 to $x \in S$, then θ does the same.

Lemma 6. *(Lee and Zhong 2020) Suppose $c = (\vee_{x \in S} x)$ is a Boolean disjunction constraint. If $e' \Rightarrow e$ where $e = \vee_{(x = v) \in \theta} v$ and $e' = \vee_{(x = v') \in \theta'} v'$, then $c\theta' \Rightarrow c\theta$.*

Note that a Boolean disjunction constraint has the property that $e \vee 0 = e$ where $e = \vee_{x \in S} x$. Thus, the *common assignment* $(x = 0)$ does not affect the validity of the sufficient condition for implied satisfaction.

Theorem 9. *Suppose we have a Boolean disjunction constraint $c = (\vee_{x \in S} x)$. If $(x_t = 0) \in \theta \cap \theta'$, then $(\vee_{(x=v') \in \theta'} v' \Rightarrow \vee_{(x=v) \in \theta} v)$ implies $(\vee_{(x=v') \in \tilde{\theta}'} v' \Rightarrow \vee_{(x=v) \in \tilde{\theta}} v)$ where $\tilde{\theta} = \theta \setminus \{x_t = 0\}$ and $\tilde{\theta}' = \theta' \setminus \{x_t = 0\}$.*

Proof. Since $(x_t = 0) \in \theta$, we have $(\vee_{(x=v) \in \theta} v) = (\vee_{(x=v) \in \tilde{\theta}} v) \vee 0 = (\vee_{(x=v) \in \tilde{\theta}} v)$. Similarly, we have $(\vee_{(x=v') \in \theta'} v') = (\vee_{(x=v') \in \tilde{\theta}'} v')$. Thus, if $(\vee_{(x=v') \in \theta'} v') \Rightarrow (\vee_{(x=v) \in \theta} v)$, then $(\vee_{(x=v') \in \tilde{\theta}'} v') = (\vee_{(x=v') \in \theta'} v') \Rightarrow (\vee_{(x=v) \in \theta} v) = (\vee_{(x=v) \in \tilde{\theta}} v)$. □

Alldifferent Constraints The *alldifferent* constraint enforces that all variables in a set T take distinct values (Régin 1994). Let $val(\theta, T) = \{v \mid (x = v) \in \theta \wedge x \in T\}$. We say that a partial assignment θ has *no duplicated values* if $\forall (x_i = v_i), (x_{i'} = v_{i'}) \in \theta$ where $x_i \neq x_{i'}$, we have $v_i \neq v_{i'}$. The sufficient condition for implied satisfaction requires that $val(\theta, T) = val(\theta', T)$.

Lemma 7. *(Lee and Zhong 2020) Suppose $c = alldifferent(T)$ where $T \subseteq X$. If $val(\theta, T) = val(\theta', T)$ and both θ and θ' have no duplicated values, then $c\theta' \Rightarrow c\theta$.*

Removing *an arbitrary common assignment* $x = v$ from θ and θ' means that the value v is removed from both $val(\theta, T)$ and $val(\theta', T)$, and the sufficient condition still holds.

Theorem 10. *If $(x = v) \in \theta \cap \theta'$, then*

1) $(val(\theta, T) = val(\theta', T)) \Rightarrow (val(\tilde{\theta}, T) = val(\tilde{\theta}', T))$

2) θ (or θ') has no duplicated values $\Rightarrow \tilde{\theta}$ (or $\tilde{\theta}'$) has no duplicated values

where $\tilde{\theta} = \theta \setminus \{x = v\}$ and $\tilde{\theta}' = \theta' \setminus \{x = v\}$.

Proof. 1) Suppose $x_t \notin T$. Since $val(\theta, T) = val(\theta', T)$, we have $val(\tilde{\theta}, T) = val(\theta, T) = val(\theta', T) = val(\tilde{\theta}', T)$. Otherwise, if $x_t \in T$, then we have $val(\theta, T) = val(\tilde{\theta}, T) \setminus \{v_t\} = val(\theta', T) \setminus \{v_t\} = val(\tilde{\theta}', T)$.
2) Since θ has no duplicated values and $\tilde{\theta} \subset \theta$, $\tilde{\theta}$ has no duplicated values; similarly for θ' and $\tilde{\theta}'$. □

Applying Common Assignment Elimination

In the following, we give examples to show CAE in action.

Example 3. *Consider the COP P in Example 1 and its associated generation problem P_g in Example 2, which consists of sufficient conditions for betterment of a separable objective implied satisfaction of a linear inequality constraint. By Theorems 5 and 8, any arbitrary assignment is commonly eliminable. Thus, we add the set of constraints $\{(x_i = v) \notin \theta \cap \theta' \mid i = 1, 2, 3$ and $v = 0, 1\}$ to P_g.*

Example 4. *Suppose we extend P in the last example with a Boolean disjunction constraint $x_1 \vee x_2$. By Theorem 9, only $(x_1 = 0)$ and $(x_2 = 0)$ fulfil the eliminability condition for $x_1 \vee x_2$, while any arbitrary assignment $x_i = v$ is commonly eliminable for other constraints. Thus, we only add the constraints $\{(x_i = 0) \notin \theta \cap \theta' \mid i = 1, 2, 3\} \cup \{(x_3 = 1) \notin \theta \cap \theta'\}$ to P_g.*

Note that eliminability of an assignment is checked by examining the form of objective and constraints. Thus, CAE can be applied mechanically by analyzing the problem model of P in only one pass.

Experimental Evaluation

We perform experiments on Xeon E7-4830 2.20GHz processors using MiniZinc 2.4.3 (Nethercote et al. 2007) to model both the benchmark problems and the corresponding dominance breaking nogood generation respectively. The back-end solver is Chuffed (Ohrimenko, Stuckey, and Codish 2009). We use six benchmark problems, and generate 20 instances for each problem configuration in order to give meaningful comparison. The following four benchmarks and associated models as well as the instance generation method are adopted from Lee and Zhong (2020): (1) *Knapsack-n* 0-1 knapsack problem with n objects, (2) *DisjKnapsack-n* disjunctively constrained 0-1 knapsack problem (Yamada, Kataoka, and Watanabe 2002) with n objects, (3) *ConcertSched-n* capacitated concert hall scheduling problem (Gange and Stuckey 2018) with n applications and 10 halls, and (4) *MaxCut-n* weighted maximum cut problem with n vertices and 0.1 edge probability. Two additional benchmarks are introduced:

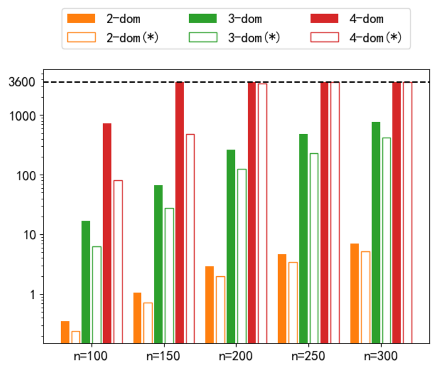
- *KnapsackSide-n* are extensions of the 0-1 knapsack problems to showcase the usefulness of Theorem 3. An instance with n items is augmented with $\lfloor 0.02n \rfloor$ table constraints, in which we randomly select three variables $x_{i_1}, x_{i_2}, x_{i_3}$ and sample each tuple from $D(x_{i_1}) \times D(x_{i_2}) \times D(x_{i_3})$ with probability 0.5. Note that side constraints can also be added to other benchmark problems.
- *PCBoard-n-m* are PC board manufacturing problems (Martin 2005) with n components and m devices, where there are *linear inequality constraints* and a *linear objective*. The model is from a public MiniZinc repository[1]. Due to the problem structure, the length of dominance breaking nogoods is at least 4.

Note that the original method of Lee and Zhong (2020) cannot handle *KnapsackSide* because of the side constraints. In addition, nogood generation for *PCBoard* was too time-consuming with the original method, and the benefits in search reduction cannot compensate the overhead. We use the search strategies specified in the public models in solving the COPs, and the default of Chuffed for the generation models. For all benchmarks, we attempt to generate all dominance breaking nogoods of length up to L without CAE (**L-dom**) as Lee and Zhong (2020) or with CAE (**L-dom(*)**) as per our proposal.

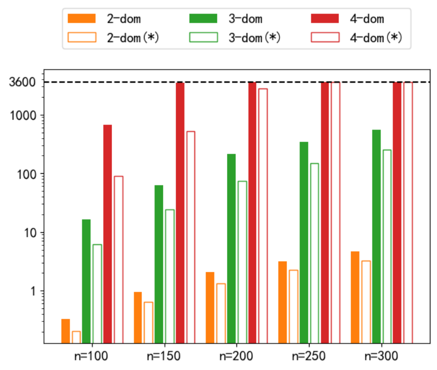
Comparison of Generation Time

We use a uniform timeout limit of 1 hour for nogood generation. Figures 1(a) to 1(f) show the time of nogood generation in *log scale*. CAE clearly reduces the nogood generation time for all benchmarks. Note that generating long nogoods without CAE could time out for some large instances, and

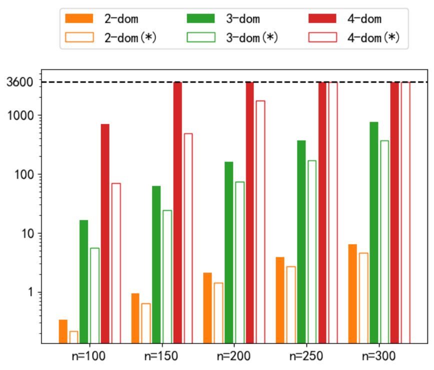
[1]https://people.eng.unimelb.edu.au/pstuckey/dominance/



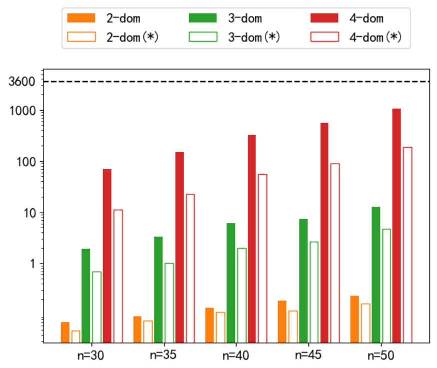
(a) Knapsack Generation Time



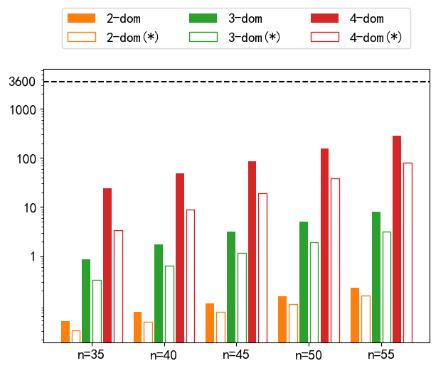
(b) DisjKnapsack Generation Time



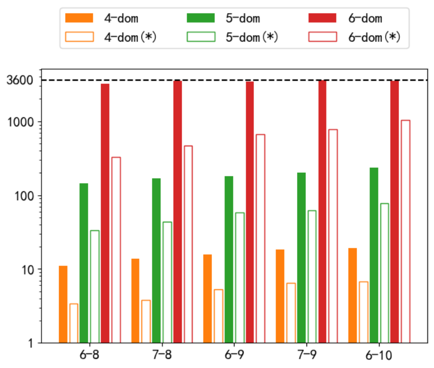
(c) KnapsackSide Generation Time



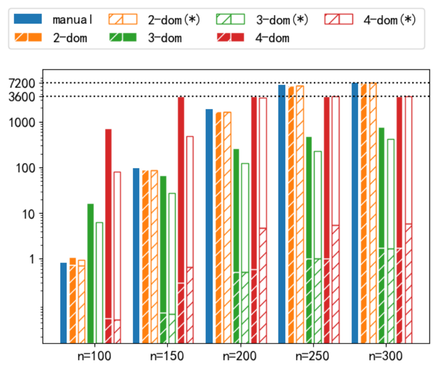
(d) ConcertSched Generation Time



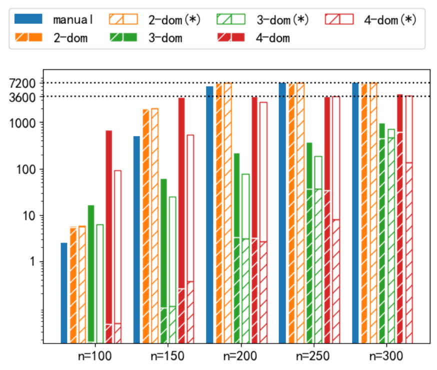
(e) MaxCut Generation Time



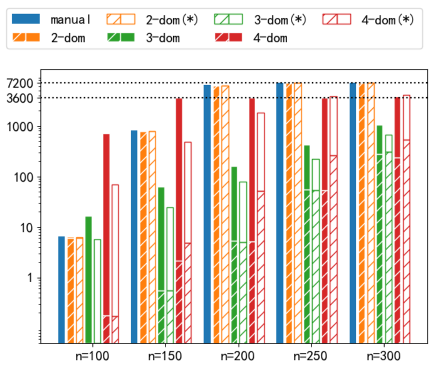
(f) PCBoard Generation Time



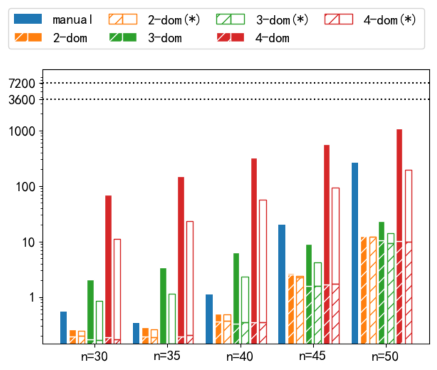
(g) Knapsack Total Time



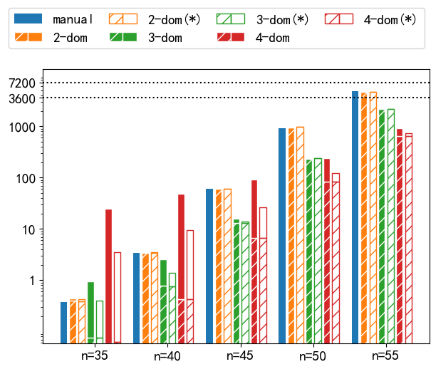
(h) DisjKnapsack Total Time



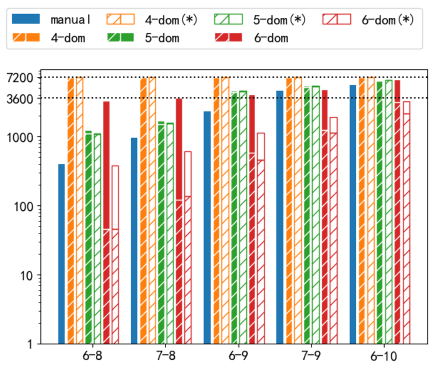
(i) KnapsackSide Total Time



(j) ConcertSched Total Time



(k) MaxCut Total Time



(l) PCBoard Total Time

Figure 1: Experimental results for comparison: Figures 1(a) to 1(f) compare the generation time, and Figures 1(g) to 1(l) compare the total time. In each subfigure, the x-axis shows different configurations, and the y-axis shows the time in log scale.

the difference between **4-dom** and **4-dom(*)** (between **6-dom** and **6-dom(*)** for *PCBoard*) can be more pronounced.

Let $t_g(L)$ and $t_g^*(L)$ denote the average time for generating nogoods of length up to L without and with CAE respectively. We also compute the *percentage decrease of generation time* $\%_g$, where $\%_g(L) = \frac{t_g(L)-t_g^*(L)}{t_g(L)}$. We only compare L-**dom** and L-**dom(*)** when both of them do not time out. The trends in *Knapsack*, *DisjKnapsack*, *KnapsackSide*, *ConcertSched* and *MaxCut* are similar. The percentage decrease is up to 37.29% between **2-dom** and **2-dom(*)**, up to 69.68% between **3-dom** and **3-dom(*)** and up to 90.08% between **4-dom** and **4-dom(*)**. As for *PCBoard*, **4-dom(*)**, **5-dom(*)** and **6-dom(*)** reduce up to 72.40%, 76.55% and 89.75% generation time compared with **4-dom**, **5-dom** and **6-dom** respectively. The benefits of CAE become more significant as we attempt to generate longer nogoods.

Comparison of Overall Performance

We study the overall performance with the total time (generation time + solving time) as the evaluation metric. The timeout for the whole solving process (nogood generation + problem solving) is set to 2 hours, while we keep the timeout for nogood generation as 1 hour. If nogood generation times out, we augment the problem model with only the nogoods generated so far. We also perform experiments using no dominance breaking constraint (**no-dom**) and using manual dominance breaking constraints (**manual**) from the literature (Chu and Stuckey 2013; Gange and Stuckey 2018) , where the timeout is set to 2 hours. We only show results of **manual** since the solving time of **no-dom** is orders of magnitude worse than that of **manual**. Figures 1(g) to 1(l) show the average solving (bars with stripe pattern) and generation time (bars without stripe pattern) in *log scale*, where the bars for generation time are stacked on the bars for solving time.

We first compare the problem solving time. Against manual dominance breaking constraints, the automatically generated dominance breaking nogoods are always stronger in search space reduction when L is large. If we compare L-**dom** and L-**dom(*)**, the problem solving times are usually the same, which is consistent with Proposition 2. The only exception is in large instances of *Knapsack*, *DisjKnapsack* and *KnapsackSide*. For these problems, both **4-dom** and **4-dom(*)** cannot finish completely within one hour, but **4-dom(*)** generates more nogoods than **4-dom**. In *Knapsack* and *KnapsackSide*, the problem solving time slightly increases due to the overhead introduced by the extra nogoods, while in *DisjKnapsack*, the benefits from the reduction of search space wins over the overhead.

In the end, the key comparison is the total time. Automatic dominance breaking outperforms **manual** in almost all benchmarks even before applying CAE. The exception is *PCBoard*, where the method of automatic dominance breaking outperforms **manual** only after CAE is applied, and the decrease of average total time is up to 2660.75 seconds for *PCBoard-9-7*. To compare the total time for L-**dom** and L-**dom(*)**, we also compute the *percentage decrease of total time* $\%_t$ for measurement, i.e. $\%_t(L) = \frac{(t_s(L)+t_g(L))-(t_s^*(L)+t_g^*(L))}{t_s(L)+t_g(L)}$, where $t_s(L)$ and $t_s^*(L)$ are the average problem solving time of COPs augmented with generated nogood of length up to L. In general, the performance gain of CAE depends on whether nogood generation takes a large part in the whole solving process. For *ConcertSched*, **2-dom** and **2-dom(*)** are usually the best, and $\%_t$ is at most 12.08%. As for *MaxCut*, **4-dom(*)** is the best for large instances, which spends up to 49.58% less than **4-dom**. The problem structures of *Knapsack*, *DisjKnapsack* and *KnapsackSide* are similar, where the problem augmented with nogoods of length up to 3 are usually the best. The percentage decrease in runtime is at least 28.44% and at most 65.57% after CAE is applied. *PCBoard* has the most to gain from CAE, which demands for longer dominance breaking nogoods. The method **5-dom** is usually the best if CAE is not applied. After applying the technique, **6-dom(*)** comes out on top, which reduces from **5-dom** by 2449.26 seconds and **6-dom** by 3034.01 seconds on average. The percentage decrease of **6-dom(*)** can be as much as 88.48% compared with **6-dom**.

Discussions

The more dominance breaking nogoods we have, the more we can prune the search space but (a) it takes time to generate the nogoods and (b) nogoods are also overheads to the propagation solver. The "optimal" nogood length for each problem varies. Even with CAE, we are able to generate nogoods with length only up to 6. Our experience is that a length of 3 or 4 is usually enough, but some experimentation may be required for different problems.

Also, we conjecture that Lee and Zhong's method hinges on a lazy clause generation solver, such as Chuffed (Ohrimenko, Stuckey, and Codish 2009), as it is good at handling (generated) nogoods. It can also combine multiple nogoods (our nogoods and/or nogoods from conflicts) to learn stronger ones, hence stronger propagation.

Concluding Remarks

Automatic dominance breaking represents an important step towards the Holy Grail (Freuder 1997). This work enlarges the class of problems that can benefit from the automation. Our specific contributions are threefold. First, we remove a restriction of the original Lee and Zhong method that all constraints must be efficiently checkable, and demonstrate that the method generates enough nogoods to speed up the solving of problems with small-scope non-EC side constraints. Second, we present techniques to avoid the generation of unnecessary nogoods and yet maintain the pruning strength of the generated nogoods. Third, our theory is backed by extensive empirical evaluation.

A future direction of work is to perform an extensive evaluation of automatic dominance breaking on public benchmarks such as the MiniZinc Challenge (Stuckey, Becket, and Fischer 2010) and the MIPLIB (Gleixner et al. 2020). Another research direction is to understand the characteristics and roles of individual (groups of) nogoods and further improve the efficiency by only generating a subset of nogoods that are most promising for search space reduction.

Acknowledgments

We are grateful to the anonymous referees of AAAI-21 for their useful comments and suggestions. We also thank the financial support by a Direct Grant from The Chinese University of Hong Kong.

References

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

Choi, C. W.; Lee, J. H. M.; and Stuckey, P. J. 2003. Propagation redundancy in redundant modelling. In *International Conference on Principles and Practice of Constraint Programming*, 229–243. Springer.

Chu, G.; and Stuckey, P. J. 2012. A generic method for identifying and exploiting dominance relations. In *Principles and Practice of Constraint Programming*, 6–22.

Chu, G.; and Stuckey, P. J. 2013. Dominance driven search. In *International Conference on Principles and Practice of Constraint Programming*, 217–229. Springer.

Freuder, E. C. 1997. In pursuit of the holy grail. *Constraints* 2(1): 57–61.

Gange, G.; and Stuckey, P. J. 2018. Sequential Precede Chain for Value Symmetry Elimination. In *International Conference on Principles and Practice of Constraint Programming*, 144–159. Springer.

Getoor, L.; Ottosson, G.; Fromherz, M.; and Carlson, B. 1997. Effective redundant constraints for online scheduling. In *The Eleventh AAAI Conference on Artificial Intelligence*, 302–307.

Gleixner, A.; Hendel, G.; Gamrath, G.; Achterberg, T.; Bastubbe, M.; Berthold, T.; Christophel, P. M.; Jarck, K.; Koch, T.; Linderoth, J.; Lübbecke, M.; Mittelmann, H. D.; Ozyurt, D.; Ralphs, T. K.; Salvagnin, D.; and Shinano, Y. 2020. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation* Accepted for publication.

Korf, R. E. 2004. Optimal rectangle packing: new results. In *The Fourteenth International Conference on Automated Planning and Scheduling*, 142–149.

Lee, J. H.; and Zhong, A. Z. 2020. Automatic dominance breaking for a class of constraint optimization problems. In *Twenty-Ninth International Joint Conference on Artificial Intelligence*, 1192–1200.

Mackworth, A. K.; and Freuder, E. C. 1985. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial intelligence* 25(1): 65–74.

Martin, R. 2005. The challenge of exploiting weak symmetries. In *International Workshop on Constraint Solving and Constraint Logic Programming*, 149–163. Springer.

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

Monette, J.-N.; Schaus, P.; Zampelli, S.; Deville, Y.; Dupont, P.; et al. 2007. A CP approach to the balanced academic curriculum problem. In *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, volume 7.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In *International Conference on Principles and Practice of Constraint Programming*, 529–543. Springer.

Ohrimenko, O.; Stuckey, P. J.; and Codish, M. 2009. Propagation via lazy clause generation. *Constraints* 14(3): 357–391.

Prestwich, S.; and Beck, J. C. 2004. Exploiting dominance in three symmetric problems. In *Fourth international workshop on symmetry and constraint satisfaction problems*, 63–70.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *The Eighth AAAI Conference on Artificial Intelligence*, 362–367.

Schulte, C.; and Stuckey, P. J. 2008. Efficient constraint propagation engines. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 31(1): 2.

Stuckey, P. J.; Becket, R.; and Fischer, J. 2010. Philosophy of the M ini Z inc challenge. *Constraints* 15(3): 307–316.

Yamada, T.; Kataoka, S.; and Watanabe, K. 2002. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal* 43(9).