Solution Dominance over Constraint Satisfaction Problems

Tias Guns[1], Peter J. Stuckey[2], and Guido Tack[2]

[1] VUB Brussels, Belgium
tias.guns@vub.be
[2] Data61 CSIRO & Monash University, Australia
pstuckey@unimelb.edu.au
guido.tack@monash.edu

Abstract. Constraint Satisfaction Problems (CSPs) typically have many solutions that satisfy all constraints. Often though, some solutions are preferred over others, that is, some solutions *dominate* other solutions. We present *solution dominance* as a formal framework to reason about such settings. We define Constraint Dominance Problems (CDPs) as CSPs with a dominance relation, that is, a preorder over the solutions of the CSP. This framework captures many well-known variants of constraint satisfaction, including optimization, multi-objective optimization, Max-CSP, minimal models, minimum correction subsets as well as optimization over CP-nets and arbitrary dominance relations.
We extend MiniZinc, a declarative language for modeling CSPs, to CDPs by introducing dominance nogoods; these can be derived from dominance relations in a principled way. A generic method for solving arbitrary CDPs incrementally calls a CSP solver and is compatible with any existing solver that supports MiniZinc. This encourages experimenting with different solution dominance relations for a problem, as well as comparing different solvers without having to modify their implementations.

1 Introduction

Constraint satisfaction has proven to be an indispensable paradigm for solving complex problems in A.I. and industry. Indeed, many such problems can be expressed as a conjunction of constraints over variables, including logical constrains, mathematical relations and sophisticated *global* constraints such as automata.

However, for many applications, the true problem to be solved is not a satisfaction problem, though satisfaction is a critical component. For example, in many cases one is interested in finding a solution that minimizes an objective function, rather than any satisfying solution. Many other settings exist in which some satisfying solutions are more interesting or preferred than others, that is, some solutions *dominate* other solutions.

We introduce *solution dominance* as a way to express such dominance relations over the solutions of a CSP. A dominance relation here defines a preorder

over the solutions. In line with Constraint Satisfaction Problems and Constraint Optimization Problems, we call the resulting problems Constraint Dominance Problems (CDPs). The goal is to find all non-dominating solutions to the CDP, that is, the Pareto optimal set. We discuss two variants, depending on whether equivalent solutions are allowed in the solution set or not.

Our work generalizes the work on preferences in SAT [7], where a strict partial order over literals is used. This captures MinOne, MaxSAT and Minimum Correction Subsets. Our work generalizes this and other works expressing preference as strict (irreflexive) partial orders, because: 1) (reflexive) preorders give us the freedom to reason both about solution sets that do or do not allow for equivalent solutions; 2) the goal is not to find all dominated (preferred) solutions, but rather all *non-dominated* solutions; hence 3) the investigated approach also capture multi-objective optimization and more. The set of non-dominated solutions is known as the Pareto frontier or the *efficient* set in multi-objective optimization [9] and our formalization is inspired by that, but can reason over arbitrary partial orders.

Preferences in SAT [7] are methodologically different from preferences expressed through Conditional Preference networks (CP-nets, [1]), because the latter requires expensive dominance checks. We investigate a novel dominance relation for CP-nets, and show that all these types of preferences can fit in the same framework and methodology. The framework can also express arbitrary solution dominance relations, including other forms of conditional preferences than CP-nets. This is motivated by recently studied data mining problems involving conditional dominance relations over CSPs [20]. A further discussion of related work is provided in Section 7.

Inspired by declarative languages for modeling CSPs [21, 10, 28] we propose an extension to the MiniZinc modeling language that enables the formulation of CDPs. The idea is to specify a *dominance nogood*, a constraint pattern that can be used whenever a solution is found during search to exclude dominated solutions in the remaining search (analogous to the well-known branch-and-bound approach for optimization). Dominance nogoods can be derived from the solution dominance relation in a principled way. We present an intuitive implementation of a generic algorithm for dominance nogoods in the MiniSearch [24] meta-search language. We explore the possibilities of this approach on a number of problems and with a range of different solvers to show the potential of such a generic approach.

2 CSPs and solution dominance

Conceptually, solution dominance can be used to solve problems of the form: find $\{X \in \mathcal{S} | \nexists Y \in \mathcal{S} ...\}$ where $\mathcal{S}$ is the set of all solutions of a CSP.

A *Constraint Satisfaction Problem (CSP)* is a triple (V, D, C) where V is a set of variables, D is a mapping from variables to a set of values, and C is a set of constraints over (a subset of) V. Constraints can represent arbitrary complex relations over the variables. A valuation X is a mapping of variables to values:

$\forall v \in V : v \mapsto D(v)$. A *solution* of CSP (V, D, C) is a valuation X that satisfies each constraint $c \in C$. We denote by $X(v)$ the value of v in solution X.

Dominance relation. A dominance relation over CSP solutions expresses when one solution dominates or is equivalent to another one. A simple example is that of a constrained optimization problem, where the optimization function defines a total (pre)order on the solutions. We follow the optimisation convention that smaller is better.

We define a dominance relation $\preceq$ as a preorder over the set of solutions of a CSP P. A preorder is a reflexive ($a \preceq a$) and transitive ($a \preceq b \wedge b \preceq c \rightarrow a \preceq c$) relation. It can be thought of as a partial order over equivalence classes. In other words, given two solutions either one dominates the other, or they are equivalent, or they are incomparable. The use of (reflexive) preorders instead of (irreflexive) partial orders allows us to discriminate between incomparable and equivalent solutions. This is also a key difference between the problem of finding all dominant/preferred solutions and all non-dominated solutions: neither of two equivalent solutions is strictly dominant, while both are non-dominant; incomparable solutions are also non-dominant.

More formally, we define the equivalence relation $X \sim Y \Leftrightarrow X \preceq Y \wedge X \succeq Y$; and the negations $\nsim$ and $\not\preceq$. Now, let $\mathcal{S}$ be the set of all solutions of a CSP, and $\preceq$ a dominance relation. We identify three possible properties for sets $A \subseteq \mathcal{S}$:

complete: every solution in $\mathcal{S}$ is dominated by or equivalent to a solution in A: $\forall X \in \mathcal{S}, \exists Y \in A : Y \preceq X$.

domination-free: the solutions in A are not dominated by any other in A, except equivalent ones: $\forall X, Y \in A : Y \not\preceq X \vee X \sim Y$. Equivalently, no $X \in A$ is strictly dominated: $\forall X \in A, \nexists Y \in A : Y \preceq X \wedge X \nsim Y$.

equivalence-free: no two solutions in A are equivalent to each other: $\forall X, Y \in A : X \nsim Y$. In preference terms, they are *indifferent* to each other.

The set of complete and domination-free solutions is unique and defined as follows: $\{X \in \mathcal{S} \mid \forall Y \in \mathcal{S}, Y \not\preceq X \vee X \sim Y\}$. In the multi-objective optimization literature [9], this set is known as the Pareto frontier or the *efficient* set. In that context it is often written in the form $\{X \in \mathcal{S} \mid \nexists Y \in \mathcal{S} : Y \preceq X \wedge X \nsim Y\}$.

In practice though, one is typically just interested in a complete and domination-free set that is also *equivalence-free*: $A \subseteq \{X \in \mathcal{S} \mid \forall Y \in \mathcal{S}, Y \not\preceq X \vee X \sim Y\}$ with $\forall X, Y \in A : X \nsim Y$. This set is not unique (it contains one arbitrary solution per equivalence class).

3 Constraint Dominance Problems

We now show how a wide range of problems that are not captured by the classical CSP framework can be expressed with a dominance relation over a CSP. Generic solving methods are discussed in the next section.

A Constraint Optimization Problem (COP) is typically defined as a tuple (V, D, C, f) where (V, D, C) is a CSP as defined before and f is a function

over a valuation of the variables V. A solution to a COP is a solution to the corresponding CSP that minimizes the function f. One is typically interested in finding one such optimal solution, that is, one solution V^* of (V, D, C) for which $\nexists V' : f(V') < f(V^*)$.

In line with COPs we define a **Constraint Dominance Problem** (CDP) as a tuple $(V, D, C, \preceq)$ where (V, D, C) is a CSP and $\preceq$ a dominance relation. Two types of queries exist for this problem (with or without equivalent solutions). We call the ***full*** *solution to the CDP* the set of complete and domination-free solutions of the CSP. A (non-unique) complete and domination-free set that is also equivalence free is simply called a *solution to the CDP*.

Optimization. Given a COP (V, D, C, f), let $\preceq_f$ be the total order corresponding to f: $X \preceq_f Y \Leftrightarrow f(X) \leq f(Y)$. Then, finding a solution to the COP corresponds to finding a solution to the CDP $(V, D, C, \preceq_f)$.

Lexicographic optimization. Let (V, D, C) be a CSP and F a set of functions $\{f_1, \ldots, f_n\}$. The goal is to find a solution that lexicographically minimizes the functions. Given the preorder $\preceq_{lex_F}$: $X \preceq_{lex_F} Y \Leftrightarrow f_1(X) < f_1(Y) \vee (f_1(X) = f_1(Y) \wedge f_2(X) < f_2(Y) \vee (\ldots \wedge f_n(X) \leq f_n(Y)))$. A solution to the CDP $(V, D, C, \preceq_{lex_F})$ is a lexicographically optimal solution.

Multi-objective optimization. The goal in multi-objective optimization is to find all Pareto optimal (non-dominated) solutions given a set of functions. Let $\preceq_F$ be the following preorder: $X \preceq_F Y \Leftrightarrow \forall_i f_i(X) \leq f_i(Y)$.

Lemma 1. *The full solution to the CDP $(V, D, C, \preceq_F)$ corresponds to the set of all Pareto optimal solutions.*

Proof.

$$
\begin{aligned}
& \{X \in S \mid \nexists Y \in S : Y \preceq_F X \wedge X \nsim_F Y\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \neg(\forall_j f_j(X) = f_j(Y))\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \exists_j f_j(X) \neq f_j(Y)\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \exists_j f_j(X) < f_j(Y)\}
\end{aligned}
$$

which is the classical definition of multi-objective optimization [9]. □

$\mathcal{X}$-minimal models. Let $\mathcal{X} \subseteq V$ be Boolean (0/1) variables of a CSP (V, C, D). Given a solution X, let $pos_{\mathcal{X}}(X) = \{v \in \mathcal{X} | X(v) = 1\}$ be the variables of $\mathcal{X}$ assigned to 1 in solution X. An $\mathcal{X}$-minimal model is a solution to the CSP such that there is no solution Y with $pos_{\mathcal{X}}(Y) \subset pos_{\mathcal{X}}(X)$. The set of all $\mathcal{X}$-minimal models is: $\{X \in S | \nexists Y \in S : pos_{\mathcal{X}}(Y) \subset pos_{\mathcal{X}}(X)\}$.
We can define the preorder $\preceq_{\mathcal{X}}$ as: $X \preceq_{\mathcal{X}} Y \Leftrightarrow \forall v \in \mathcal{X} : X(v) \leq Y(v)$, that is, for each variable of $\mathcal{X}$ an assignment to *false* (domain value 0) is preferred over *true* (domain value 1).

Lemma 2. *The full solution to the CDP $(V, D, C, \preceq_{\mathcal{X}})$ corresponds to the set of $\mathcal{X}$-minimal models.*

Proof. Using the same rewriting as for multi-objective optimization, we obtain that the set of non-dominated solutions is:

$$\begin{aligned}
& \{X \in S \mid \nexists Y \in S : Y \preceq_{\mathcal{X}} X \wedge X \nsim_{\mathcal{X}} Y\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_{v \in \mathcal{X}} Y(v) \leq X(v) \wedge \exists_{v \in \mathcal{X}} X(v) \neq Y(v)\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_{v \in \mathcal{X}} pos(Y \cap \{v\}) \subseteq pos(X \cap \{v\}) \\
& \qquad \wedge \exists_{v \in \mathcal{X}} pos(X \cap \{v\}) \neq pos(Y \cap \{v\})\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : pos_{\mathcal{X}}(Y) \subseteq pos_{\mathcal{X}}(X) \wedge pos_{\mathcal{X}}(X) \neq pos_{\mathcal{X}}(Y)\}
\end{aligned}$$

which equals $\{X \in S | \nexists Y \in S : pos_{\mathcal{X}}(Y) \subset pos_{\mathcal{X}}(X)\}$. □

The concept of minimal models can be extended to non-Boolean CSPs by defining a total order $\leq_v$ over the possible values of each variable $v \in V$, in line with $X(v) \leq Y(v)$.

Weighted (partial) MaxCSP. Given a CSP (V, D, C) and a function $g : C \to \mathbb{R}$ that represents the weight of a constraint. Let X be a valuation which need not be a solution to the CSP. The total weight of X is the sum of the weights of the constraints that are satisfied by X: $w(X) = \sum_{c \in C, c(X)=true} g(c)$. The goal is to find an assignment to V that maximizes this weight. As our dominance relation is over solutions of a CSP, we define a new CSP (V', D', C') as follows: a set of $|C|$ new Boolean variables B is added to V: $V' = V \cup B$, and each (soft) constraint is replaced by a reified version: $C' = \{B_c \to c | \forall c \in C\}$. Let preorder $\preceq_g$ be $X \preceq_g Y \Leftrightarrow w'(g, X) \geq w'(g, Y)$, with $w'(g, X) = \sum_c g(c) * X(B_c)$, then, the weighted MaxCSP problem (V, D, C) given g is equivalent to the CDP $(V', D', C', \preceq_g)$. Because $\preceq_g$ is a total order, this is equivalent to a COP over w'.

Valued CSPs. A *valued* CSP with annotated constraints [26] has a *valuation structure* $(E, \oplus, \leq_v, \bot, \top)$. Each constraint is mapped to a problem specific value in E. The values are aggregated using operator $\oplus$, and $\leq_v$ defines a total over E. Because of the total order, a similar encoding to that for weighted MaxCSP can be obtained.

Maximally satisfiable subsets. A maximally satisfiable subset $M \subseteq C$ of (V, D, C) is such that (V, D, M) is satisfiable and adding any other constraint leads to unsatisfiability: $\forall c \in C \setminus M : (V, D, M \cup c)$ is unsatisfiable. Dually, we call $C \setminus M$ a minimal correction subset.

Applying the same transformation of (V, D, C) to (V', D', C') as for MaxCSP problems, the goal is to find solutions X with set $pos_B(X) = \{c \in C | X(B_c) = 1\}$ of active constraints such that no additional constraints in C can be added to it: $\nexists Y \in S : pos_B(Y) \supset pos_B(X)$. This corresponds to finding all *B-maximal* models. In line with minimal models, we can define the preorder $X \preceq_{MSS} Y \Leftrightarrow \forall c \in C : X(B_c) \geq Y(B_c)$ and corresponding CDP $(V', D', C', \preceq_{MSS})$.

CP-nets. A CP-net is an acyclic directed graph over a set of variables V [1]. Each node in a CP-net corresponds to a variable V_i and has a conditional preference table $CPT(V_i)$. A CPT associates with each possible partial valuation a of the *parent* variables in the graph, a strict total order $<_i^a$ over

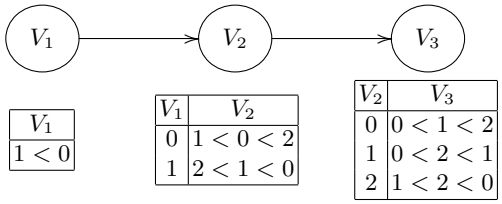

Fig. 1. CP-net example over 3 variables.

the values of V_i (consistent with the rest of this paper, $x < y$ means x is preferred over y). Figure 1 shows an example.
A CP-net induces a set of *preference rankings* that are consistent with all CPTs, where a preference ranking is a total ordering over all complete valuations of V. Traditionally, dominance in a CP-net is defined in terms of its preference rankings: o dominates o' if in all of the preference rankings of the CP-net o is ordered before o'. Even for binary-valued CP-nets, the complexity of a dominance check for an arbitrary network is NP-hard [1].
An easier to compute query is the *ordering query* [1]: given two valuations o and o', is there a CPT where for all ancestor variables o and o' have the same value and o is preferred to o' according to the CPT: $\exists v \in V : (\forall w \in Ancestor_{\mathcal{N}}(v), o(w) = o'(w) \wedge o <_{CPT_{\mathcal{N}}(v)} o')$? If so, there must exist a preference ordering where $o < o'$ and hence this implies that o' does not dominate o. It only *implies* non-dominance though (sufficient but not necessary), meaning that there may not exists such a CPT and yet o' does not dominate o. This is because only CPTs are checked for which all **ancestors** have the same value in o and o'; hence there may be a CPT with $Ancestor_{\mathcal{N}}(v) \neq Parent_{\mathcal{N}}(v)$ that induces preference rankings in which $o < o'$ but the CPT is not checked because of the ancestor condition.
We hence propose the following *weaker* form of CP-net dominance:

Definition 1. *Given CP-net $\mathcal{N}$, a valuation o locally dominates o' iff for all CPTs: when its **parents** have equal value in o and o' but the CPT variable does not, then o must be preferred to o' by the CPT;*

$$o \prec_{\mathcal{N}} o' \Leftrightarrow \forall v \in V : (\ (\forall w \in Parents_{\mathcal{N}}(v), o(w) = o'(w) \wedge o(v) \neq o'(v)) \rightarrow o <_{CPT_{\mathcal{N}}(v)} o'\ ) \quad (1)$$

Note that there is always at least one v with corresponding CPT active, because a CP-net is acyclic and hence there is at least one node with an empty parent set.

Lemma 3. *Local dominance in a CP-net is a necessary but not a sufficient condition for traditional (preference ranking-based) dominance.*

Proof. All preference rankings have to agree with all CPTs. Hence, if o dominates o' then for all applicable (parent variables equal, current variable not)

CPTs: $o <_{CPT_{\mathcal{N}}(v)} o'$, so local dominance must necessarily be true. However, if $o \not\prec_{\mathcal{N}} o'$ then by negation there must exists a node v with applicable CPT for which $o \not<_{CPT_{\mathcal{N}}(v)} o'$ and hence because of $o(v) \neq o(v')$: $o' <_{CPT_{\mathcal{N}}(v)} o$. However, it may still be the case that o dominates o', because of an interaction between the grandparent variables and CPTs such that no preference ranking actually uses this entry of $CPT_{\mathcal{N}}(v)$. This cannot be observed from the *local* CPTs directly, but must be verified through a (NP-hard) dominance check over the preference rankings.

Definition 2. *Because each CPT in a CP-net encodes strict relations, two valuations can only be equivalent if they are identical:* $Y \sim_{\mathcal{N}} X \equiv Y = X$. *We hence define* $Y \preceq_{\mathcal{N}} X \equiv Y \prec_{\mathcal{N}} X \vee Y = X$.

Local dominance can be checked in time linear in the number of CPTs in the CP-net. We can hence realistically use it to enumerate all non-locally-dominated solutions of a CP-net by solving CDP $(V, D, C, \preceq_{\mathcal{N}})$. The resulting solution will be an over-approximation of the actual set of traditional non-dominated solutions. Should the application demand it, one could still filter the resulting set by post-processing with the NP-hard traditional dominance check.

Domain-specific dominance relations. We showcase the need for domain-specific dominance relations in data mining. Increasingly, constraint programming is used for data mining problems such as searching for *patterns* that appear *frequently* in a database [13]. A pattern can be a set of items, a sequence or another structure such as a graph [14]. A pattern is *frequent* if it is a subpattern of sufficiently many objects in the database. The problem is typically encoded as a set of constraints that define the pattern type, and that define when a pattern is frequent. One solution to this CSP is then one frequent pattern.

However, there are a number of pattern mining settings that do not fit the classical CSP framework, most notably *closed* and *maximal* patterns, *relevant* patterns and *skyline* patterns. Using the concept of dominance though, these problems can be modeled declaratively and combined with arbitrary constraints [20].

For example, a frequent pattern is *maximal* if there is no other frequent pattern that is a *superpattern* of this pattern. Let $\sqsubseteq$ represent the subpattern relation, e.g. subset, subsequence or subgraph relation depending on the pattern type. The general dominance relation for maximal patterns is: $X \preceq_{maximal} Y \equiv X \sqsupseteq Y$. Let S be the set of all frequent patterns, then the full solutions is $\{X \in S \mid \nexists Y \in S : Y \sqsupseteq X \wedge X \nsim Y\} \equiv \{X \in S \mid \nexists Y \in S : Y \sqsupset X\}$, the set of all *maximally* large patterns such that no other frequent pattern is a superpattern of it. For patterns represented by a set I, the subpattern relation is the subset relation over I, hence; $\{X \in S \mid \nexists Y \in S : pos_I(Y) \supset pos_I(X)\}$. Note that this is equivalent to an $\mathcal{X}$-maximal model with $\mathcal{X} = I$.

Closed frequent patterns have the weaker condition that there should not be a superpattern with the same frequency: dominance relation $X \preceq_{closed}$

$Y \equiv X \sqsupseteq Y \wedge freq(X) = freq(Y)$. The resulting full solution set is the set of all frequent patterns for which no superpattern exists that has the same frequency: $\{X \in S \mid \nexists Y \in S : Y \sqsupset X \wedge freq(Y) = freq(X)\}$, this is true for all pattern types. An *algebra* inspired by databases is used in [20] to show how a number of other pattern mining problems can be expressed as a combination of a constraint algebra and a dominance algebra. The framework we propose here focuses on one CSP and one dominance relation, which is sufficient for most settings.

4 Search and dominance nogoods

The main task that we consider is to find a solution to the constraint dominance problem $(V, D, C, \preceq)$, that is, a complete, domination-free and optionally equivalence-free set of solutions. Our methodology is to solve the problem through a chain of constraint satisfaction problems, similar in spirit to [7].

Let $\mathcal{O}(V, D, C)$ be an oracle that returns a satisfying solution to the CSP (V, D, C) or fails if no such solution exists.

Complete. the following algorithm returns a complete solution to a CSP (V, D, C) with dominance relation $\preceq$, using oracle $\mathcal{O}$:

Algorithm 1 search$(V, D, C, \preceq, \mathcal{O})$:

```
1: A := ∅
2: while S := O(V, D, C) do
3:     A := A ∪ {S}
4:     C := C ∪ {S ⋠ V ∨ S ∼ V}
5: end while
6: return A
```

Note how on line 4, a constraint is added over variables V such that any (future) assignment to these variables may not be strictly dominated by the found solution S. As the constraint set C is monotonically increasing, the oracle can be *incremental* in that it can continue its search from where its last solution was found; the result is a branch-and-bound style algorithm where the bound is represented by $S \not\preceq V \vee S \sim V$.

Let $\langle S_1, S_2, \ldots, S_n \rangle$ be the sequence of solutions as found by the oracle in Algorithm 1.

Theorem 1. *The set A returned by Algorithm 1 is a complete set:* $\forall X \in (V, D, C)$, $\exists Y \in A : Y \preceq X$.

Proof. The set is complete: let P_x be the CSP (V, D, C) that yields solution S_x, let $\mathcal{S}(P_x)$ be the set of *all* solutions of P_x. P_1 is the original CSP and hence $\mathcal{S}(P_1)$ contains all solutions of the CSP and this set is complete. The subsequent

P_2 only forbids solutions that are dominated and not equivalent to S_1, hence $\{S_1\} \cup \mathcal{S}(P_2)$ is also complete as any solution is either dominated by S_1 or in $\mathcal{S}(P_2)$. By induction, any set $\{S_1, S_2, \ldots, S_x\} \cup \mathcal{S}(P_{x+1})$ is complete. Hence, $S = \{S_1, \ldots, S_n\} = \{S_1, \ldots, S_n\} \cup \mathcal{S}(P_{n+1})$ is complete as the stopping criterion on line 2 dictates that $\mathcal{S}(P_{n+1}) = \emptyset$. □

Complete and equivalence-free. Algorithm 1 can easily be modified to return a complete and equivalence-free set. In this case, line 4 has to be changed to $C := C \cup \{S \not\preceq V\}$. This set is equivalence-free as now any solution to $\mathcal{S}(P_{x+1})$ cannot be equivalent to S_x. The set is still complete as only equivalent solutions are additionally removed by the modification, and hence for each solution X in (V, D, C) it will still be the case that $\exists Y \in A : Y \preceq X$.

Domination-free. Any solution S_y found after S_x cannot be strictly dominated by S_x, as this is explicitly forbidden by the added constraint. That means that if the oracle enumerates the solutions from most to least preferred according to the preorder (e.g. first assign variables to 1, then to 0 for MaxCSP), then the complete and equivalence-free set is also domination-free. This is the approach used by [7].

However, if we assume no order on the solutions found by the oracle, it is possible to find, with $y > x$, an S_y that strictly dominates S_x: $S_y \preceq S_x \wedge S_y \nsim S_x$. We can remove these by doing a *backwards pass* over the solutions in which we check for each $S_y, S_x, y > x$ that $S_y \not\preceq S_x \vee S_y \sim S_x$ and if not we drop S_x. Note that we reuse oracle $\mathcal{O}$ for this though it merely has to *check* whether a fixed assignment to the variables satisfies the constraints.

The same procedure can be applied to a complete and equivalence-free set to make it domination-free; this set is already equivalence-free so the process will only remove the strictly dominated solutions.

From dominance relation to dominance nogood. We refer to the constraint added on line 4 of Algorithm 1 as the dominance nogood. It can be derived from the preorder $\preceq$ in a principled way. We will demonstrate this for a number of earlier examples.

Since one is often interested in a complete, domination-free and equivalence-free set, we will demonstrate it for the equivalence-free dominance nogood $S \not\preceq V$. We denote the relation representing the dominance nogood as $D(S, V)$. It can often be obtained by negating the logical definition of the preorder $\preceq$. The dominance nogood with equivalence $S \not\preceq V \vee S \sim V$ can be obtained following similar methods as well, perhaps more easily in the equivalent form $S \not\preceq V \vee V \preceq S$. Recall that V is the set of variables of the CSP and S is a previously found solution.

Optimization. For dominance relation $X \preceq_f Y \Leftrightarrow f(X) \leq f(Y)$ the dominance nogood is $D(S, V) \Leftrightarrow \neg(f(S) \leq f(V)) \Leftrightarrow f(S) > f(V) \Leftrightarrow f(V) < f(S)$. This guarantees that every new assignment to V will have a smaller score than the previously found solution S.

Lexicographic optimization. The dominance relation is $X \preceq_{lex_F} Y \Leftrightarrow f_1(X) < f_1(Y) \vee (f_1(X) = f_1(Y) \wedge (f_2(X) < f_2(Y) \vee (\ldots \wedge f_n(X) \leq f_n(Y))))$. The negation is the dominance nogood: $D(S,V) \Leftrightarrow f_1(S) \geq f_1(V) \wedge (f_1(S) \neq f_1(V) \vee (f_2(S) \geq f_2(V) \wedge (\ldots \vee f_n(S) > f_n(V))))$. Using the observation that $(A \geq B) \wedge ((A \neq B) \vee Z) \Leftrightarrow (B < A) \vee ((B = A) \wedge Z)$ we obtain $D(S,V) \Leftrightarrow f_1(V) < f_1(S) \vee (f_1(V) = f_1(V) \wedge (f_2(V) < f_2(S) \vee (\ldots \wedge f_n(V) < f_n(S))))$. One could also use a `lex_less` global constraint.

Multi-objective optimization . The dominance relation $X \preceq_F Y \Leftrightarrow \forall_i f_i(X) \leq f_i(Y)$ has the dominance nogood $D(S,V) \Leftrightarrow \exists_i f_i(S) > f_i(V)$. Recall that as in the previous examples, this is the dominance nogood to obtain the *equivalence-free* set of complete and domination-free solutions.

$\mathcal{X}$-minimal models We have $X \preceq_{\mathcal{X}} Y \Leftrightarrow \forall v \in \mathcal{X} : X(v) \leq Y(v) \Leftrightarrow pos_{\mathcal{X}}(X) \subseteq pos_{\mathcal{X}}(Y)$ and dominance nogood $D(S,V) \Leftrightarrow \exists v \in \mathcal{X} : S(v) > V(v)$.

Minimal correction subsets We noted earlier that minimal correction subsets is dual to finding the maximal satisfiable subsets, which corresponds to the problem of finding all maximal models of (V', D', C') with $X \preceq_{MSS} Y \Leftrightarrow \forall c \in C : X(B_c) \geq Y(B_c)$. The corresponding dominance nogood is $D(S,V) \Leftrightarrow \exists c \in C : S(B_c) < V(B_c)$.

CP-nets For CP-nets we have $D(S,V) \Leftrightarrow S \neq V \wedge S \not\prec_{\mathcal{N}} V \;\Leftrightarrow\; S \neq V \wedge \exists v : (\forall w \in Parents(v), S(w) = V(w) \wedge S(v) \neq V(w) \wedge V <_{CPT(v)} S)$.

Each line in a CPT is mutually-exclusive, so the relation that V is more preferred to S according to $CPT(V_i)$, $V <_{CPT(v)} S$, can be formalized using a disjunction over all CPT entries. The preference in each entry can be modeled using implications. For example, for the first entry of $CPT(X_3)$ in Figure 1 as follows: $V_2 = 0 \wedge S_2 = 0 \wedge V_3 \neq S_3 \wedge ((S_3 = 2 \rightarrow (V_3 = 1 \vee V_3 = 0)) \vee (S_3 = 1 \rightarrow V_3 = 0))$. Indeed, the parents have to take a specific (identical) value, and in that case if $S_3 = 2$ then V is preferred only if V_3 takes value 1 or 0; if $S_3 = 1$ then V_3 must be 0 to be preferred. Because of the mutual exclusivity and because S is a solution of which we know the values, when posting the dominance nogood, at most one entry per CPT will be part of the logical formula.

Complexity. Each time the oracle $\mathcal{O}$ is called, it has to solve an NP-hard CSP (V, D, C) in general. Furthermore, there can be an exponential number of non-dominated solutions and hence calls to the oracle: there exists a search order such that the algorithm has to enumerate all solutions to find all non-dominant ones. Brafman *et al* [3] show that the simpler problem of computing the *next solution* in a CSP or preference problem is in general also NP-hard, although there are some special cases (like tree CSPs) where it can be easier.

Note that since the constraint system is monotonically increasing, any learned nogoods from previous runs (from a learning solver [22], or from restarts [18]) are valid, hence each subsequent call to the solve oracle $\mathcal{O}$ can take advantage of them.

5 Integration in a modeling language

Any constraint solver that supports incrementally adding constraints and retrieving the next solution can implement Algorithm 1. From the modeling perspective, we describe how MiniZinc [21], a modeling language for CSPs and COPs, can be extended to handle CDPs as new modeling primitive.

Instead of extending MiniZinc to express dominance relations, we instead add **syntax for dominance nogoods**, for two reasons: 1) users can explicitly specify either an equivalence-free dominance nogood, or a dominance nogood with equivalence; 2) we found it more intuitive to declare an *invariant* for the search (e.g. minimization as $f(V) < f(S)$ where S is a previously found solution), rather then declaring when a previous solution dominates or is equivalent to a new one (e.g. $f(S) \leq f(V)$).

MiniZinc has keywords to define variables, constraints, predicates and a minimal search specification. We add the keyword `dominance_nogood` for specifying dominance nogoods. Furthermore, the built-in MiniSearch [24] function `sol(X)` can be used to refer to the value of variable X in a previously found solution. Hence, the dominance nogood for a minimization problem: $D(S, V) \Leftrightarrow f(V) < f(S)$ can be expressed as:

```
dominance_nogood f(V) < f(sol(V));
```

with V some array of variables of the accompanying CSP.

Here is an example for finding minimal correction subsets (note the `not` in the output statement, where we convert the maximal satisfiable subset to a minimal correction subset).

```
array [int] of var bool: B;
constraint B[1] -> ...;
dominance_nogood exists(i in index_set(B))(B[i] < sol(B[i]));
output ["MCS:"] ++ ["\(i) " | i in index_set(B)
                              where not fix(B[i])];
```

5.1 Solving

Algorithm 1 can be specified in a straightforward way using the recently introduced MiniSearch language [24]. MiniSearch is an extension of MiniZinc with support for specifying meta-level search heuristics. The integration with MiniZinc allows any existing FlatZinc solver to be used by MiniSearch, and hence also for solving CDPs.

At the language level, a declaration `dominance_nogood e` is simply translated into a predicate declaration `predicate post_dng() = e` that posts the dominance nogood. Algorithm 1 is then specified as follows in MiniSearch:

```
solve search dominance_search;
function ann: dominance_search() =
  repeat( if next() then
            commit() /\ print() /\ post_dng()
          else break endif );
```

The `dominance_search` function repeatedly queries a black-box solver for the next solution (`next()`), and if a solution is found, it remembers it (`commit()`), prints it, and then posts the dominance nogood before continuing to search for the next solution. In MiniSearch, `next()` can either call an external solver process through a file based interface, restarting the search from scratch for each call; or use an incremental C++ API that permits adding constraints during the search.

6 Experiments

We evaluate the viability of the framework and the opportunities of modeling constraint dominance problems in a declarative solver-independent language. We do not aim to beat the state-of-the-art on any one specific task, as they typically employ specialized bounds or additional inference mechanisms. The experiments below evaluate different categories of dominance nogoods and show that they can be handled through the generic framework presented in this paper, including the novel setting of optimizing over CP-nets using generic CSP solvers.

The experiments use MiniSearch with the following solvers: Gecode 4.4.0 (gecode), or-tools v2015-09 (ortools), Opturion CPX 1.0.2 (optcpx) and Chuffed b776ac2 (chuffed). Gecode and or-tools are classical depth-first search CP solvers, while Opturion CPX and Chuffed are lazy clause generation solvers [22]. All solvers are called by MiniSearch as external processes using the standard file-based interface. Gecode additionally supports the direct, incremental C++ API (gecode-api).

6.1 MaxCSP

We use benchmarks from the 2008 XCSP competition, MaxCSP *with globals* category. Table 1 shows a comparison of MiniSearch with different solvers on a selection of instances. For each solver, we compare a model where no variable order is given ('free') with the specification of a most-to-least preferred strategy over the B variables only ('ord', e.g. assign to 1 before 0). The latter forces the solvers to first consider all constraints, then all but one (arbitrary) constraint, and so on.

We can see in Table 1 that providing the search order often leads to improved runtimes, but not always (`quasigrp` for gecode, `cabinet` for chuffed). gecode is slower than the incremental API approach gecode-api in case an order is given; when doing free search, the restarts of the file-based approach seem to improve runtime for `q13` (and others, not shown). The remaining solvers seem to handle this task pretty well, especially optcpx. For a rough comparison, in the 2008 competition the `quasigrp` and `latinSq` instances were also solved within seconds, however runtimes of 600+ seconds were reported for `cabinet` and 40+ seconds for `q13`.

Table 1. MaxCSP runtimes in seconds, — timed out after 30 min.

Instance	gecode-api		gecode		ortools		chuffed		optcpx	
	free	ord	free	ord	free	ord	free	ord	free	ord
cabinet-5570	—	0.9	—	—	36	0.2	257	—	3.9	0.3
cabinet-5571	—	0.9	—	—	36	0.2	257	—	3.9	0.4
latinSq-dg-3_all	0.2	0.1	0.5	0.3	0.1	0.1	0.2	0.3	0.1	0.1
latinSq-dg-4_all	0.6	0.9	0.8	6.8	0.5	1.3	0.5	13	0.6	0.3
quasigrp4-4	46	—	—	—	4.5	—	3.8	18	1.4	7.7
quasigrp5-4	0.4	1651	1158	—	1.1	—	1.6	5.4	1.6	1.3
q13-1110973670	479	1.1	32	0.9	540	0.7	635	43	11	7.5
q13-1111219348	569	1.1	32	1.3	385	0.9	641	72	8.8	7.0

6.2 Multi-objective

We consider traveling saleseman problems where two different costs are given between any two cities, e.g. duration and fuel cost. We report on the generation of all complete and equivalence-free solutions. The straightforward *backward pass* needed to make the set domination-free is omitted for reasons of simplicity. The instances are from the Oscar repository [23].

In Table 2 we compare the MiniSearch approach with different solvers and Oscar [23], which has an efficient dedicated propagator for multi-objective optimisation [15]. The first three lines use the first-fail variable ordering used in Oscar, the last three use a *max-regret* ordering over the distance variables, as found in MiniZinc's TSP models. The gecode-api results indicate that file-based restarts lead to much slower solving times. The number of intermediate solutions also has a big influence on runtime, as using a better variable order leads to both smaller solution sets and smaller runtimes.

6.3 CP-net

The following experiments consider a variant of the Photo problem, where the goal is to find an ordering of friends such that the number of preferences regarding

Table 2. Runtime and number of solutions (forward pass) for multi-objective optimization; — indicates time out after 30 min; n.a. that the search strategy was not supported

Instance	gecode-api		gecode		ortools		chuffed		oscar	
	time	sols	time	sols	time	sols	time	sols	time	sols
ren10	0.5	108	7.5	108	6.8	108	38	105	2.3	110
ren15	368	949	—	545	–	565	—	343	61	891
ren20	—	998	—	382	–	392	—	381	—	—
ren10-mg	1.8	41	2.8	41	1.3	45	5	38	n.a.	n.a.
ren15-mg	14	135	247	135	541	145	—	128	n.a.	n.a.
ren20-mg	—	925	—	292	—	294	—	171	n.a.	n.a.

Table 3. CP-net photo-like setting, runtimes and number of solutions (forward pass), — indicates timeout after 30 min.

Instance	gecode-api		gecode		ortools		chuffed		optcpx	
	time	sols	time	sols	time	sols	time	sols	time	sols
10v-4p-1	109	76	126	78	144	81	110	61	192	112
10v-4p-2	14	14	46	43	17	16	7	7	20	18
10v-4p-3	31	29	51	43	14	12	80	57	78	61
10v-4p-4	397	673	782	285	496	331	547	378	—	753
10v-4p-5	15	68	101	289	34	95	14	42	24	76

whom to stand next to for a group photo is maximized. We here consider the case that preferences are supplied as a CP-net: each person indicates a number of people (parents in the CP-net) and their preferences considering the locations of these people. Such a CP-net can be partial, i.e., it can contain disconnected components, which our method can handle without any modification.

We randomly generated CP-nets with n people ($5 \leq n \leq 20$) and for each person between 0 and k parents (sampled uniformly per person, $1 \leq k \leq 10$). The induced order in the CPTs corresponds to preferring smaller average distance to the parent(s). We again report only the forward pass of computing all complete and equivalence-free solutions. Larger n and larger k lead to larger CP-nets and runtimes, though that depends very much on the actual CP-net generated.

Table 3 shows results on 5 different networks generated with $n = 10$ and $k = 4$. No variable/value ordering strategy was imposed. The number of solutions found clearly has an influence on runtime, where the number of solutions is not only specific to the problem at hand, but also to the search order chosen by the solver. We expect a method that uses the expensive (traditional) dominance checks to perform worse.

7 Related Work and discussion

As discussed in the introduction, most related is the work on preferences in SAT [25, 7], where a preference can be defined over individual literals. They identify *preference formulas* for these tasks, which correspond to dominance nogoods, and use incremental SAT solving. Our framework generalizes this to a wider range of tasks and different solving technology.

As with any generic method, one cannot expect to obtain the most efficient method for each of the covered tasks. Indeed, specialized methods have been developed for MaxSAT [6], minimum correction subsets [19] and X-minimal models [29] that can be more efficient than a SAT with preferences approach, but only for their specific task. For multi-objective optimization in CSPs, specialized propagation algorithms exist that filter the search space more effectively [11, 15]. Similarly for other forms of preference [17].

Nevertheless, recent applications of data mining using SAT [16] and CSP [20] demonstrate the need for generic methods for handling novel *solution dominance* settings, for example involving *conditional* dominance relations.

Many works in CP-nets focus on consistency and dominance testing [8, 1, 2]. A branch-and-bound style algorithm for finding all non-dominated solutions given additional constraints has been studied [2]; it uses expensive (PSPACE complete [12]) dominance checks in case the ordering query returns false. Furthermore the search (variable order) in their method is driven by the CP-net's structure. In contrast, our method proposes a novel dominance relation that is cheap to evaluate and provides an over-approximation; it can be used with any existing solver and for any variable order.

A recent extension of Answer Set Programming [4] covers some of the tasks in this paper too (no CP-nets or domain-specific relations), but within the stable model semantics. They provide a language extension for expressing preference relations with a preference type (e.g. less, subset, pareto) and preference elements (the variables). Our language extension is closer to the original dominance relation which can make it easier to specify domain-specific dominance nogoods.

The concept of *dominance* is also used in different contexts in the constraint programming community. Dominance breaking [5] for COPs creates constraints that, given a mapping σ, prevent the finding of solutions θ such that $\sigma(\theta)$ is a better solution of the COP. They can drastically improve solving performance. [5] rely on a notion of dominance relation that applies to all valuations and not just solutions, so that sub-trees can be pruned during the search. In that sense they are complementary to the solution dominance we consider in this paper. Indeed an interesting direction for further work is to extend dominance breaking to arbitrary solution dominance problems.

8 Conclusions

We introduced the concept of *solution dominance*, where the dominance relation is a preorder over the solutions of a CSP. We call the resulting problems Constraint Dominance Problems, and this captures single/lexicographic/multi-objective optimization, X-minimal models, weighted MaxCSP, minimum correction subsets as well as a novel dominance relation for reasoning over CP-nets, as well as other dominance relations. We provide a natural and declarative extension to MiniZinc for specifying Constraint Dominance Problems, based on MiniSearch.

Preferences and (solution) dominance have a history in CP research, as discussed above. Two directions for future work hence emerge: 1) which other preference [17] or dynamic solving settings [27] fit the solution dominance framework; and 2) what specialised solving methods that have been investigated for one category (e.g. MaxCSP or Multi-objective) can also be applied to other settings? From a modeling perspective one can also ask the question whether it can be automatically detected from a dominance nogood specification, that a more specialised algorithm could be used (e.g. for lexicographic or bi-objective optimisation).

References

1. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. (JAIR) 21, 135–191 (2004)
2. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: Preference-based constrained optimization with cp-nets. Computational Intelligence 20, 137–157 (2004)
3. Brafman, R., Rossi, F., Salvagnin, D., Venable, K., Walsh, T.: Finding the next solution in constraint- and preference-based knowledge representation formalism. In: Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR2010). pp. 425–433. AAAI Peess (2010)
4. Brewka, G., Delgrande, J.P., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1467–1474 (2015)
5. Chu, G., Stuckey, P.J.: Dominance breaking constraints. Constraints 20(2), 155–182 (2015)
6. Davies, J., Bacchus, F.: Exploiting the power of mip solvers in maxsat. In: Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings. pp. 166–181 (2013)
7. Di Rosa, E., Giunchiglia, E., Maratea, M.: Solving satisfiability problems with preferences. Constraints 15(4), 485–515 (Oct 2010)
8. Domshlak, C., Brafman, R.I.: Cp-nets - reasoning and consistency testing. In: In Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning. pp. 121–132. Morgan Kaufmann (2002)
9. Ehrgott, M.: Multicriteria optimization. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag (2000)
10. Frisch, A., Harvey, W., Jefferson, C., Hernández, B.M., Miguel, I.: Essence: A constraint language for specifying combinatorial problems. Constraints 13(3), 268–306 (2008)
11. Gavanelli, M.: An algorithm for multi-criteria optimization in csps. In: Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002. pp. 136–140 (2002)
12. Goldsmith, J., Lang, J., Truszczynski, M., Wilson, N.: The computational complexity of dominance and consistency in cp-nets. J. Artif. Intell. Res. (JAIR) 33, 403–432 (2008)
13. Guns, T.: Declarative pattern mining using constraint programming. Constraints 20(4), 492–493 (2015)
14. Guns, T., Paramonov, S., Negrevergne, B.: On declarative modeling of structured pattern mining. In: AAAI Workshop on Declarative Learning Based Programming, Phoenix, Arizona USA, 12-13 February 2016 (2016)
15. Hartert, R., Schaus, P.: A support-based algorithm for the bi-objective pareto constraint. In: Brodley, C.E., Stone, P. (eds.) AAAI 2014. pp. 2674–2679. AAAI Press (2014)
16. Jabbour, S., Sais, L., Salhi, Y.: The top-k frequent closed itemset mining using top-k SAT problem. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III. pp. 403–418 (2013)

17. Junker, U.: Preference-based search and multi-criteria optimization. Annals of Operations Research 130(1-4), 75–115 (2004)
18. Lecoutre, C., Sais, L., Tabary, S., Vidal, V.: Nogood recording from restarts. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007. pp. 131–136 (2007)
19. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013 (2013)
20. Négrevergne, B., Dries, A., Guns, T., Nijssen, S.: Dominance programming for itemset mining. In: 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013. pp. 557–566 (2013)
21. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a standard CP modelling language. In: CP. LNCS, vol. 4741, pp. 529–543. Springer (2007)
22. Ohrimenko, O., Stuckey, P., Codish, M.: Propagation via lazy clause generation. Constraints 14(3), 357–391 (2009)
23. OscaR Team: OscaR: Scala in OR (2015), available from `https://bitbucket.org/oscarlib/oscar`
24. Rendl, A., Guns, T., Stuckey, P.J., Tack, G.: MiniSearch: A solver-independent meta-search language for minizinc. In: Pesant, G. (ed.) CP2 015. LNCS, vol. 9255, pp. 376–392. Springer (2015)
25. Rosa, E.D., Giunchiglia, E., Maratea, M.: A new approach for solving satisfiability problems with qualitative preferences. In: ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings. pp. 510–514 (2008)
26. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes. pp. 631–639 (1995)
27. Ugarte Rojas, W., Boizumault, P., Loudni, S., Crémilleux, B., Lepailleur, A.: Mining (soft-) skypatterns using dynamic csp. In: Simonis, H. (ed.) Integration of AI and OR Techniques in Constraint Programming. pp. 71–87. Springer International Publishing, Cham (2014)
28. Van Hentenryck, P.: The OPL optimization programming language. MIT Press (1999)
29. Zohary, R.B.E..: An incremental algorithm for generating all minimal models. Artificial Intelligence 169(1), 1 – 22 (2005)